TESIS – KI142502

ALGORITMA TIME-BASED ALPHA MINER UNTUK MEMODELKAN PROSES BISNIS DAN PENGOPTIMASIAN MENGGUNAKAN SISTEM MANUFAKTUR FLEKSIBEL DI TERMINAL PETIKEMAS

YUTIKA AMELIA EFFENDI
NRP. 05111650010033

DOSEN PEMBIMBING
Prof. Drs.Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D
NIP: 195908031986011001

PROGRAM MAGISTER
RUMPUN MATA KULIAH MANAJEMEN INFORMASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN KOMUNIKASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

2017

THESIS – KI142502

TIME-BASED ALPHA MINER OF BUSINESS PROCESS MODEL AND THE OPTIMIZATION USING FLEXIBLE MANUFACTURING SYSTEM IN PORT CONTAINER TERMINAL

YUTIKA AMELIA EFFENDI
NRP. 05111650010033

SUPERVISOR
Prof. Drs.Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D
NIP: 195908031986011001

MAGISTER PROGRAM
INFORMATION MANAGEMENT
DEPARTMENT OF INFORMATICS
FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2017

Tesis disusun untuk memenuhi salah satu syarat memperoleh gelar
Magister Komputer (M.Kom.)
di
Institut Teknologi Sepuluh Nopember Surabaya

oleh:
Yutika Amelia Effendi
NRP. 05111650010033

Dengan judul :
ALGORITMA TIME-BASED ALPHA MINER UNTUK MEMODELKAN PROSES BISNIS DAN PENGOPTIMASIAN MENGGUNAKAN SISTEM MANUFAKTUR FLEKSIBEL DI TERMINAL PETIKEMAS

Tanggal Ujian : 29-12-2017
Periode Wisuda : 2017 Gasal

Disetujui oleh:

Prof.Ir.Drs.Ec. Riyanarto Sarno, M.Sc, Ph.D
NIP. 195908031986011001
(Pembimbing 1)

Tohari Ahmad, S.Kom, MIT, Ph.D
NIP. 197505252003121002
(Penguji 1)

Dr. Ir. Raden Venantius Hari Ginardi, M.Sc
NIP. 196505181992031003
(Penguji 2)

Bagus Jati Santoso, S. Kom., Ph. D.
NIP. 1986201711051
(Penguji 3)

Dekan Fakultas Teknologi Informasi dan Komunikasi,

Dr. Agus Zainal Arifin, S.Kom., M.Kom.
NIP. 19720809 199512 1 001

Algoritma Time-based Alpha Miner untuk Memodelkan Proses Bisnis dan Pengoptimasian Menggunakan Sistem Manufaktur Fleksibel di Terminal Petikemas

Nama mahasiswa	: Yutika Amelia Effendi
NRP	: 05111650010033
Pembimbing	: Prof. Drs.Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D.

ABSTRAK

Manajemen proses bisnis dilakukan untuk mendapatkan hasil yang diinginkan dalam waktu dan biaya yang optimum. Waktu yang optimum dapat dicapai dengan menambah sumber daya. Sedangkan, biaya yang optimum dapat dicapai dengan mengurangi sumber daya. Oleh karena itu, diperlukan sistem yang dapat mengoptimasi proses bisnis. Namun, optimasi hanya dapat bekerja apabila ada data model proses bisnis. Sehingga juga diperlukan sistem yang dapat mengenali proses bisnis.

Proses bisnis bisa diperoleh dengan menggunakan teknik *process discovery* yang bekerja dengan menggali relasi dari data log. Relasi tersebut adalah *sequence*, paralel (XOR, OR, dan AND). *Process discovery* yang sudah ada dapat menggali relasi paralel (XOR, OR, dan AND), *sequence*, perulangan (*loop)*, *non-free choice*, dan *invisible task*. Selain itu, sebagian besar *process discovery* yang sudah ada juga menggunakan *single timestamp* untuk menemukan model proses. Di dalam penelitian yang telah dilakukan oleh peneliti-peneliti sebelumnya, aktivitas dapat diparalelkan dengan cara mengetahui hubungan resiprokal (misal: aktivitas A berelasi *sequence* kepada aktivitas B dan sebaliknya) antaraktivitas untuk menggali relasi paralel dalam data log. Misal *trace* AB, BA untuk paralel antara aktivitas A dan B. Namun, tidak dikaitkan dengan kondisi yang dibutuhkan untuk memparalelkan kedua aktivitas di data log.

Untuk memparalelkan sebuah proses bisnis, aktivitas yang independen harus diidentifikasi terlebih dahulu, seperti aktivitas manakah dari proses bisnis yang dapat dilakukan bersamaan. Tingkat paralelisme tertinggi dicapai jika jumlah aktivitas yang diidentifikasi sebagai independen dapat dimaksimalkan. Secara umum, identifikasi ini didasarkan pada waktu dan tempat eksekusi aktivitas, aktivitas dapat di paralelisasi jika aktivitas pada entitas simulasi yang sama dijalankan dalam urutan *timestamp*. Untuk meningkatkan tingkat paralelisme, kami mengusulkan sebuah pendekatan baru yang mengidentifikasi kriteria independensi lain: Jika dua aktivitas pada entitas simulasi yang sama mengakses item data yang sama dengan cara yang berbeda, mereka dapat dieksekusi secara paralel.

Pada Tesis ini akan diusulkan kondisi yang diperlukan untuk memparalelkan dua kegiatan di dalam data log dan mengembangkan sistem yang dapat memodelkan relasi proses bisnis secara otomatis dan dapat mengoptimasi biaya dan waktu sekaligus.

Optimasi biaya dan waktu dilakukan dengan mempertimbangkan sumber daya (*machine)* dengan menggunakan Sistem Manufaktur Fleksibel (FMS) untuk memperoleh jumlah total mesin dapat berganti fungsi dalam satu hari dan satu bulan serta *Goal Programming* yang digunakan untuk mengoptimasi biaya dan waktu dari setiap departemen sehingga menghasilkan nilai waktu dan biaya yang optimum.

Hasil eksperimen menunjukkan bahwa algoritma Modified Time-based Alpha Miner dapat menemukan model proses dengan benar serta relasi paralel AND, OR dan XOR, sementara algoritma original Alpha hanya dapat mengkategorikan relasi paralel menjadi AND dan XOR. Setelah model proses ditemukan, dengan menggunakan Sistem Manufaktur Fleksibel, Departemen *Behandle* dan Karantina dapat dieksekusi paralel dengan merubah fungsi mesin RTGC menjadi HT Truck. Dengan menggunakan kakas bantu LEKIN dan metode *First Come First Serve* (FCFS), hasil penjadwalan di Terminal Petikemas dapat diketahui berdasarkan jenis *container*nya. Lalu, hasil optimasi dengan *Goal programming* adalah waktu maksimum setiap aktivitas yang diminimalkan menjadi rata-rata durasi eksekusi per aktivitas dan biaya yang dapat dihemat dari waktu maksimum tersebut.

Kata kunci: optimasi waktu dan biaya; *double timestamp data log*; sistem manufaktur fleksibel; FMS; proses bisnis; *overlap time;* algoritma Alpha; *completeness*; interval waktu; *goal programming*

Time-based Alpha Miner of Business Process Model and the Optimization Using Flexible Manufacturing System (FMS) in Port Container Terminal

Name	: Yutika Amelia Effendi
Student ID	: 05111650010033
Supervisor	: Prof. Drs.Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D.

ABSTRACT

Bussiness process management is used to produce product within optimized time and cost. The optimized time can be achieved by adding resource; whereas, the optimized cost can be achieved by decreasing resource. Therefore, it is needed a system to optimize business process. However, the optimization only works if business process model available. Hence, it is also needed a system to discover business process.

Business process can be obtained by using process discovery technique that works by mining relation from event log. The relation which can be obtained is sequence and parallel (XOR, OR, and AND). Existing process discovery can discover the parallel relations (XOR, OR, and AND), sequence, loop, non-free choice, and invisible task. In addition, most existing discovery processes use single timestamp to discover the process model. In previous study done by researchers, activity can be paralleled by knowing the reciprocal relationship (e.g. activity A has sequence relation to activity B and vice versa) to mine the parallel relation in the event log, e.g. trace AB, BA to parallel between activity A and B. However, it does not have necessary condition to parallelize the two activities in the event log.

To parallelize simulations of business process, independent activities have to be identified, which can be executed concurrently. The highest level of parallelism is achieved if the number of activities identified as independent is maximized. Traditionally, this identification is based on time and location of activities, only allowing parallelization if activities on the same simulation entity are executed in timestamp order. To increase the level of parallelism, we propose a novel approach investigating another criterion for independence: If two activities on the same simulation entity do not access the same data items in a conflicting manner, they can as well be executed in parallel.

In this Thesis research, we will propose the necessary conditions to parallelize two activities in the event log and develop a system that can model business processes automatically and can optimize the cost and time at once. Cost and time optimization is done by considering the resources (machine) by using Flexible Manufacturing System (FMS) to obtain the total number of machines can change the function in one day and one

month and Goal Programming which is used to optimize cost and time of each department so as to generate value optimum time and cost.

The experimental results show that Modified Time-based Alpha Miner algorithm can find process models correctly and gateway parallel AND, OR and XOR, while the original Alpha Miner algorithm can only categorize parallel relations into AND and XOR. Once the modeling process is found, using the Flexible Manufacturing System, the Behandle and Quarantine departments can be executed parallel by changing the function of the RTGC engine to HT Truck. By using LEKIN tools and First Come First Serve (FCFS) method, the scheduling result in container terminal can be known based on container type. Then, the optimization result with Goal programming is the maximum time each activity is minimized to the average duration of execution per activity and the cost can be saved from that maximum time.

Keyword: time and cost optimization; double timestamp event log; parallel business process; flexible manufacturing system; overlap time; Alpha algorithm; completeness; time-based interval; goal programming

KATA PENGANTAR

Alhamdulillahirabbil'alamin. Puji dan syukur penulis panjatkan kehadirat Allah SWT atas berkat, rahmat dan hidayah-Nya, penyusunan Tesis ini dapat diselesaikan. Tesis ini dibuat sebagai salah satu syarat dalam menyelesaikan Program Studi Magister di Institut Teknologi Sepuluh Nopember Surabaya. Penulis menyadari bahwa Tesis ini dapat diselesaikan karena dukungan dari berbagai pihak, baik dalam bentuk dukungan moral dan material.

Melalui kesempatan ini dengan kerendahan hati penulis mengucapkan terima kasih dan penghargaan setinggi-tingginya kepada semua orang untuk semua bantuan yang telah diberikan, antara lain kepada:

1. Papa Jon Effendi dan Mama Fatmawita tercinta yang tiada henti selalu mendukung anaknya, tetap sabar mendengar keluhannya, selalu mendoakan anaknya yang terbaik dan selalu menjadi panutan yang baik.
2. Adek jagoan satu-satunya, Aditya Pernanda Effendi yang cerewet tetapi tetap dapat diandalkan dalam segala hal, sehingga dapat membangkitkan semangat penulis untuk mengerjakan tesis ini.
3. Bapak Prof. Drs.Ec. Ir. Riyanarto Sarno, M.Sc., Ph.D. selaku pembimbing yang senantiasa memberikan arahan dan bimbingan kepada penulis. Semoga Allah SWT senantiasa merahmati bapak dan keluarga.
4. Kementrian Pendidikan dan Kebudayaan RI atas beasiswa yang telah diberikan kepada penulis sehingga penulis dapat melanjutkan pendidikan master di Teknik Informatika ITS
5. Sahabat penulis (Madis, Asri, Julia, mbak Rury, Arya), 'Tim CAKDY' (Afina, Kelly, Charis, Dewi) dan 'Geng Sosialita' (mbak Dita, mbak Ulum, mbak Santi, Adhi, Ipul, Icang, Ridwan, mas Ilmi) yang selalu mendukung, menyemangati, membantu dan mendengarkan suka duka selama proses pengerjaan Tesis.
6. Rey dan Rizal dari kelas Audit Sistem S1, terima kasih atas bantuan yang telah diberikan
7. Seluruh dosen S2 Teknik Informatika yang telah memberikan ilmu dan pengetahuan kepada penulis selama menempuh studi.

8. Bapak Amin, selaku Manajer IT PT Terminal Petikemas Surabaya yang sudah memberi izin agar data PT TPS dapat digunakan dan diolah pada tesis ini.
9. Serta semua pihak yang turut membantu penulis dalam menyelesaikan tesis ini.

Akhirnya dengan segala kerendahan hati penulis menyadari masih banyak terdapat kekurangan pada Tesis ini. Oleh karena itu, segala tegur sapa dan kritik yang sifatnya membangun sangat penulis harapkan demi kesempurnaan Tesis ini. Penulis berharap bahwa perbuatan baik dari semua orang yang dengan tulus memberikan kontribusi terhadap penyusunan Tesis ini mendapatkan pahala dari Allah. Aamiin Alluhamma Aamiin.

Surabaya, Desember 2017

Penulis

DAFTAR ISI

(*halaman ini sengaja dikosongkan*)

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR ISTILAH

Process mining	Penambangan proses
Process discovery	Penemuan proses
Event log	Data log
Input	Masukan
Output	Keluaran
Sequence	Berurutan
Parallel	Paralel
Workflow	Alur kerja
Concurrent	Dikerjakan dalam waktu bersamaan
Invisible task	Aktivitas yang tidak terlihat
Non-free choice	Pilihan tidak bebas
Short loops	Perulangan pendek
Trace	Lintasan
Timestamp	Informasi waktu
Single Timestamp	Informasi waktu tunggal
Double Timestamp	Informasi waktu ganda
Case	Kasus
Activity	Aktivitas
Split	Terpecah
Join	Terhubung
Noise	Eror acak
Enterprise	Perusahaan
Resource	Pelaku aktivitas
Machine learning	Pembelajaran mesin
Length two loop	Perulangan ganda
Length one loop	Perulangan tunggal

Long loop	Perulangan panjang
Short loop	Perulangan pendek
Completeness	Suatu kondisi dimana data log menyimpan seluruh perilaku yang bisa dieksekusi pada proses bisnis
Activity Lifespan	Selisih waktu akhir dan mulai sebuah aktivitas
Case	Suatu kasus tertentu yang ada pada data log.
ProM	Kakas bantu yang digunakan untuk menganalisa dan menggambarkan proses berdasarkan data

BAB I.
PENDAHULUAN

Pada bab ini akan dijelaskan mengenai beberapa hal dasar dalam pembuatan proposal penelitian yang meliputi latar belakang, perumusan masalah, tujuan, manfaat, kontribusi penelitian, dan batasan masalah.

1.1 Latar Belakang

Manajemen proses produksi diperlukan untuk mendapatkan hasil yang diinginkan dalam waktu dan biaya yang terbatas. Proses produksi yang paling optimum adalah proses yang menghasilkan produk dalam waktu dan biaya seminimal mungkin. Waktu minimum bisa diraih dengan menambahkan mesin atau pekerja. Namun, penambahan sumber daya ini membutuhkan biaya. Sementara itu, biaya minimal bisa diraih dengan mengurangi sumber daya. Oleh karena itu, manajemen memerlukan formula yang menghasilkan produk pada waktu dan biaya optimal. Kasus ini disebut "*time and cost optimization* (TCO)". "Manajemen waktu proyek" dan "manajemen biaya proyek" adalah dua inti domain yang paling penting dalam memperoleh biaya efektif. Masalah optimasi biaya adalah masalah yang merupakan crossover dari dua domain inti.

Berbagai model optimasi yang berbeda telah diusulkan untuk memecahkan TCO, salah satunya dengan menggunakan Sistem Manufaktur Fleksibel (FMS). Model FMS merupakan kemampuan perusahaan untuk merespon secara efektif perubahan yang terjadi, baik yang terjadi di internal (operasi) perusahaan, maupun di eksternal lingkungan perusahaan. Pada FMS setiap job guna memproduksi sesuatu, mempunyai beberapa alternatif jalur mesin–mesin untuk menyelesaikannya. Sistem penanganan material pada FMS harus dikontrol komputer untuk menentukan alternatif jalur job tadi secara otomatis. Disiplin antrian yang digunakan biasanya adalah *First Come First Serve* (FCFS), *Last Come First Serve* (LCFS) atau prioritas. Optimasi biaya dan waktu dilakukan dengan mempertimbangkan sumber daya (*machine)* dengan menggunakan Sistem Manufaktur Fleksibel (FMS) untuk memperoleh jumlah total mesin dapat berganti fungsi dalam satu hari dan satu bulan serta *Goal Programming* yang digunakan untuk mengoptimasi biaya dan waktu dari setiap departemen sehingga menghasilkan nilai waktu dan biaya yang optimum.

Ada banyak model data yang dapat merepresentasikan proses bisnis. Data log dengan *single timestamp* adalah salah satu data yang sering ada di dunia nyata. Dengan melakukan modifikasi, data log dengan *single timestamp* dapat diubah menjadi data log dengan *double timestamp*. Data log dengan *double timestamp* memiliki data aktivitas yang ada dalam proses bisnis, waktu pelaksanaan aktivitas di data log dalam setiap kasus (waktu mulai eksekusi aktivitas, waktu selesai eksekusi aktivitas dan waktu tunggu antar aktivitas), dan data biaya setiap aktivitas dalam setiap kasus dalam log peristiwa. Oleh karena itu dalam Tesis ini akan diusulkan metode untuk menghitung data log dengan

double timestamp yang digunakan untuk optimasi waktu dan biaya yang mempertimbangkan *resource* dengan menggunakan Sistem Manufaktur Fleksibel (FMS) dan Goal Programming.

Selain itu, untuk mengoptimalkan proses produksi diperlukan Standar Operasional Prosedur (SOP). SOP adalah serangkaian instruksi tertulis yang menjadi standar proses produksi suatu perusahaan. SOP optimal dapat diketahui dari data log perusahaan yang diproses dengan menggunakan teknik *process discovery* (Aalst, 2010). Data log adalah dokumen yang menyimpan catatan kegiatan proses produksi perusahaan.

Process discovery memiliki beberapa algoritma, diantaranya yaitu algoritma *Alpha, Alpha+, Alpha++*, algoritma genetika (Borja Vazquez-Barreiros, Manuel Mucientes, Manuel Lama, 2015), *Heuristics Miner* (Sofie De Cnudde, Jan Claes, Geert Poels, 2014), *Temporal Activity-based Process Discovery* (Riska A. Sutrisnowati, Hyerim Bae, Dongha Lee, Kim Minsoo, 2014), dan lain-lain. Masing-masing algoritma memiliki kelebihan dan kekurangan tersendiri. Dengan demikian, algoritma perlu dimodifikasi untuk mengatasi kekurangannya. Algoritma *Alpha++*, yang merupakan peningkatan algoritma *Alpha* dan *Alpha+*, dapat menemukan *loop* dan *non-free choice*, namun masih memiliki *noise*, *incompleteness*, dan *conditional OR*. Algoritma Heuristics Miner yang menggunakan informasi waktu dapat mengatasi *noise*, menemukan *loop*, dan paralel OR.

Selain permasalahan dalam penggalian perilaku *conditional OR*, terdapat permasalahan lain yaitu *incompleteness*. *Incompleteness* disebabkan karena data log hanya terdiri dari sedikit data. Oleh karena itu, setiap algoritma memiliki gagasan *completeness* suatu log yang digunakan untuk penemuan model proses. Gagasan ini berbeda-beda bergantung pada metode yang digunakan algoritma. Algoritma alpha, alpha+, dan alpha++ menggunakan prinsip hubungan sekuensial dalam penggalian modelnya. Sedangkan algoritma *Temporal Activity-based Process Discovery* menggunakan informasi waktu sebagai dasar penemuan model prosesnya. Namun, penggunaan *double timestamp* lebih baik dalam mengatasi *incompleteness* dan penemuan model proses daripada *single timestamp* (Riska A. Sutrisnowati, Hyerim Bae, Dongha Lee, Minsoo Kim, 2014).

Meskipun *process discovery* yang ada dapat menemukan *loop*, relasi paralel (OR, XOR dan OR), mengatasi *noise* dan *incompleteness*, teknik *process discovery* tidak memiliki kondisi yang diperlukan untuk memparalelkan kedua aktivitas dalam data log selain melihat hubungan resiprokal. Untuk memparalelkan sebuah proses bisnis, aktivitas yang independen harus diidentifikasi terlebih dahulu, seperti aktivitas manakah dari proses bisnis yang dapat dilakukan bersamaan. Tingkat paralelisme tertinggi dicapai jika jumlah aktivitas yang diidentifikasi sebagai independen dapat dimaksimalkan. Secara umum, identifikasi ini didasarkan pada waktu dan tempat eksekusi aktivitas, aktivitas dapat di paralelisasi jika aktivitas pada entitas simulasi yang sama dijalankan dalam urutan *timestamp*. Untuk meningkatkan tingkat paralelisme, kami mengusulkan sebuah pendekatan baru yang mengidentifikasi kriteria independensi lain: Jika dua aktivitas pada

entitas simulasi yang sama mengakses item data yang sama dengan cara yang berbeda, mereka dapat dieksekusi secara paralel.

Dalam penelitian Tesis ini, solusi yang diajukan adalah memberikan kondisi yang diperlukan untuk memparalelkan dua atau lebih aktivitas di dalam data log dan memodifikasi algoritma Alpha sehingga dapat menemukan relasi *sequence*, relasi paralel (AND, OR, XOR) dengan informasi waktu (*double timestamp*) dan mempertimbangkan gagasan *completeness* dari data log. Algoritma yang dimodifikasi ini akan diuji studi kasus PT. Terminal Petikemas Surabaya. Kemudian, model proses yang ditemukan akan digunakan untuk tujuan optimasi waktu dan biaya yang mempertimbangkan *resource* (*machine*) dengan menggunakan Sistem Manufaktur Fleksibel (FMS) dan *Goal programming.*

1.2 Perumusan Masalah

Rumusan masalah yang diangkat dalam penelitian Tesis ini adalah sebagai berikut:

1) Bagaimana memberikan kondisi yang diperlukan untuk memparalelkan dua aktivitas atau lebih di dalam data log?
2) Bagaimana memodifikasi algoritma *Alpha* sehingga dapat menemukan relasi *sequence* dan relasi paralel (AND, OR, XOR) dengan informasi waktu dan mempertimbangkan gagasan *completeness* dari data log?
3) Bagaimana melakukan *scheduling* dengan Sistem Manufaktur Fleksibel?
4) Bagaimana cara memperoleh biaya dan waktu proses perusahaan paling optimal dengan *Goal programming*?

1.3 Tujuan dan Manfaat

1.3.1 Tujuan

Tujuan dari penelitian Tesis ini antara lain:

1) Untuk memberikan kondisi yang diperlukan untuk memparalelkan dua atau lebih aktivitas di dalam data log
2) Untuk memodifikasi algoritma *Alpha* sehingga dapat menemukan relasi *sequence* dan relasi paralel (AND, OR, XOR) dengan informasi waktu dan mempertimbangkan gagasan *completeness* dari data log
3) Untuk melakukan *scheduling* dengan Sistem Manufaktur Fleksibel
4) Untuk memperoleh biaya dan waktu proses perusahaan paling optimal dengan *Goal programming*
5) Membangun sistem yang dapat menemukan model proses dari data log dan mendefinisikan *gateway* paralel dari relasi paralel yang ditemukan

1.3.2 Manfaat

Manfaat dari penelitian Tesis ini adalah untuk mendefinisikan kondisi yang diperlukan untuk memparalelkan dua atau lebih aktivitas di dalam data log, memodifikasi

algoritma Alpha sehingga dapat menemukan relasi *sequence* dan relasi paralel (AND, OR, XOR) dengan informasi waktu dan mempertimbangkan gagasan *completeness* dari data log, melakukan *scheduling* dengan Sistem Manufaktur Fleksibel, memperoleh waktu dan biaya optimal dengan *Goal programming,* dan membangun sistem yang dapat menemukan model proses dari data log.

1.4 Kontribusi Penelitian

Kontribusi dari penelitian pada Tesis ini adalah analisis, modifikasi algoritma dan membangun aplikasi yang dapat menemukan model proses dari data log, yang meliputi beberapa metode:

1. Menentukan kondisi untuk memparalelkan dua atau lebih aktivitas di dalam proses bisnis
2. Modifikasi algoritma *Alpha* sehingga dapat menemukan relasi *sequence* dan relasi paralel (AND, OR, XOR) dengan informasi waktu dan mempertimbangkan gagasan *completeness* dari data log
3. Melakukan *scheduling* mesin agar berjalan optimal dan tidak memiliki waktu *waiting time* terlalu lama
4. Memperoleh biaya dan waktu proses produksi perusahaan paling optimal dengan menggunakan *Goal programming*.

1.5 Batasan Masalah

Batasan masalah pada penelitian ini adalah:

1. Pemodelan proses bisnis yang digunakan dalam bentuk Petri Net yang digambarkan dengan menggunakan kakas bantu WoPeD
2. Data masukan sistem dalam format excel (.xlsx atau .csv)
3. Untuk memodelkan proses bisnis dan menemukan relasi *sequence* dan paralel, bahasa pemrograman yang digunakan adalah Java, sedangkan untuk mendefinisikan *gateway* paralel model proses bahasa yang digunakan adalah Python, untuk *scheduling* mesin dengan FMS kakas bantu yang digunakan adalah LEKIN, sedangkan kakas bantu yang digunakan untuk optimasi waktu dan biaya dengan *Goal programming* adalah LINGO
4. Data log yang akan digunakan untuk memodelkan proses bisnis (*process discovery)* dan optimasi waktu dan biaya mengandung Case ID, aktivitas, waktu, biaya, dokumen dan mesin yang digunakan
5. *Process discovery* dengan modifikasi Algoritma Alpha digunakan untuk memodelkan proses bisnis dari data log
6. Sistem Manufaktur Fleksibel (FMS) digunakan untuk melakukan *scheduling* mesin yang menjalankan aktivitas dengan metode FCFS (*First Come First Serve*)
7. Optimasi waktu dan biaya aktivitas dengan menggunakan *Goal Programming*
8. Relasi model proses hasil *discovery* adalah seluruh relasi *sequence* dan paralel.

BAB II
KAJIAN PUSTAKA DAN DASAR TEORI

Bab ini menjelaskan penelitian sebelumnya dan bahan pendukung terkait penelitian yang dilakukan untuk mencapai tujuan penelitian dalam menemukan kondisi atau aturan untuk memparalelkan dua atau lebih aktivitas, algoritma *Alpha,* data log dengan *double timestamp, completeness*, dan optimasi waktu dan biaya dari model proses.

2.1 Kajian Pustaka

Bagian ini menjelaskan penelitian sebelumnya yang berkaitan dengan *process discovery*, paralel aktivitas, dan optimasi waktu dan biaya.

2.1.1 Algoritma berbasis Waktu dalam *Process Discovery*

Algoritma Heuristics Miner adalah salah satu algoritma yang paling banyak digunakan dalam *process discovery* (Cnude S, 2014). Meskipun algoritma asli Heuristics Miner hanya dapat menemukan rangkaian dan hubungan paralel AND dan XOR (Andrea Burattin, Alessandro Sperduti, 2010), namun dengan modifikasi, algoritma ini dapat menemukan relasi paralel OR. Algoritma ini menggunakan data log dengan *single timestamp* (hanya memiliki waktu mulai) untuk menemukan model proses. Selain itu, algoritma lainnya yang juga menggunakan data log dengan *single timestamp* yaitu *Temporal Activity-based Process Discovery* (Riska A. Sutrisnowati, Hyerim Bae, Dongha Lee, Kim Minsoo). Tabel 2.1 menjelaskan setiap penelitian menggunakan algoritma yang menggunakan data log dengan informasi waktu sebagai algoritma yang digunakan dalam proses penemuan model.

Tabel 2.1 Penelitian menggunakan algoritma berbasis waktu dalam *process discovery*

No	Algoritma/ Metode	Penjelasan algoritma/Metode, Kelebihan dan Kekurangan
1	Algoritma *Heuristics Miner* (A.J.M.M. Weijters, W.M.P. van der Aalst, dan A.K. Alves de Medeiros)	Algoritma yang diusulkan adalah algoritma *Heuristics Miner* asli yang digunakan untuk penemuan model proses dari data log. Kelebihan: Penelitian ini menggambarkan algoritma *Heuristics Miner* dari awal sampai pada tahap penemuan model proses. Dengan menggunakan banyak data log, algoritma ini membuktikan bahwa dengan menerapkan prinsip frekuensi informasi aktivitas dan *single timestamp* dapat menemukan model proses dengan benar. Kekurangan: Algoritma ini hanya bisa menemukan model proses dengan relasi sequence dan relasi paralel (XOR dan AND) dan *long loop* dengan data log

		dengan *single timestamp* (hanya memiliki waktu mulai).
2	*Flexible Heuristics Miner* (FHM) (A.J.M.M. Weijters, J.T.S Ribeiro)	Algoritma yang diusulkan adalah algoritma *Heuristics Miner* yang digunakan untuk menemukan model proses dari data log. Penelitian ini mengembangkan algoritma *Heuristics Miner* sebagai bagian dari kakas bantu ProM. Kelebihan: Penelitian ini menggambarkan algoritma *Heuristics Miner* dari awal sampai pada tahap penemuan model proses. Penelitian ini menambahkan algoritma *Heuristics Miner* sebagai bagian dari kakas bantu ProM 5. Kekurangan: Algoritma ini hanya bisa menemukan model proses dengan relasi *sequence* dan relasi paralel (XOR dan AND) dan *long loop* dengan data log dengan *single timestamp* (hanya memiliki waktu mulai).
3	Meningkatkan kualitas Algoritma *Heuristics Miner* di kakas bantu ProM 6.2. (Sofie De Cnuddea, Jan Claes, Geert Poelsa)	Algoritma yang diusulkan adalah algoritma *Heuristics Miner* yang digunakan untuk menemukan model proses dari data log. Penelitian ini mengembangkan algoritma *Heuristics Miner* sebagai bagian dari kakas bantu ProM 6.2 dan meningkatkan kinerja algoritma ini. Kelebihan: Penelitian ini menggambarkan algoritma *Heuristics Miner* dari awal sampai pada tahap penemuan model proses. Studi ini menambahkan algoritma *Heuristics Miner* sebagai bagian dari kakas bantu ProM 6.2. Kemudian mengukur kompatibilitas, validitas, dan kelengkapan algoritma ini dengan memastikan data log yang digunakan tidak mengandung error. Kekurangan: Algoritma ini hanya bisa menemukan model proses dengan relasi *sequence* dan relasi paralel (XOR dan AND) dan *long loop* dengan data log dengan *single timestamp* (hanya memiliki waktu mulai).
4	*Heuristics Miner* untuk Interval Waktu (Andrea Burattin and Alessandro Sperduti)	Algoritma yang diusulkan adalah algoritma *Heuristics Miner* yang digunakan untuk menemukan model proses dari data log. Penelitian ini mengembangkan algoritma *Heuristics Miner* menggunakan data log

		dengan informasi *double timestamp* (waktu mulai dan waktu selesai). Kelebihan: Penelitian ini menggambarkan algoritma *Heuristics Miner* dari awal sampai tahap penemuan model proses dari data log menggunakan informasi *double timestamp* (waktu mulai dan waktu selesai). Kekurangan: Algoritma ini hanya bisa menemukan model proses dengan relasi *sequence*, relasi paralel (XOR dan AND) dan *long loop*.
5	*Modified Time-based Heuristics Miner Algorithm* (Riyanarto Sarno, Yutika Amelia Effendi, Fitrianing Haryadita)	Algoritma yang diusulkan adalah modifikasi algoritma *Heuristics Miner* yang digunakan untuk penemuan model proses dari data log. Penelitian ini mengembangkan algoritma *Heuristics Miner* menggunakan data log dengan informasi *double timestamp* (waktu mulai dan waktu selesai) dan interval waktu sebagai dasar *process discovery*. Data log yang diuji adalah proses produksi benang. Kemudian mengukur *fitness* dan validitas algoritma ini dengan memastikan data log tidak mengandung error. Kelebihan: Algoritma modifikasi ini dapat menemukan model proses yang mengandung relasi paralel OR dan *short loop* dengan memanfaatkan informasi waktu dari data log. Hasil *fitness* dan validitas model proses yang ditemukan lebih baik daripada algoritma *Heuristics Miner* yang asli. Kekurangan: Algoritma yang diusulkan tidak mampu mengatasi masalah *incompleteness*.
6	Penemuan Model Proses berdasarkan *Activity Lifespan* (Riska A. Sutrisnowati, Hyerim Bae, Dongha Lee, Minsoo Kim).	Metode ini menggunakan *activity lifespan* dan prinsip *non-linear dependency* untuk menemukan model proses. Dalam penelitian ini didefinisikan hubungan antar aktivitas di data log, yaitu *before and meets*, *overlaps*, *contains*, *is-finished-by*, *equals*, and *starts*. Kemudian, dengan definisi ini, penulis mengusulkan teorema yang lebih kompleks yaitu hubungan sequence dan paralel. Kelebihan: Metode ini dapat menemukan model proses yang mengandung relasi *sequence*, paralel (AND dan XOR), *noise*, *completeness*, dan penggunaan informasi *single timestamp*.

		Kekurangan: Penulis tidak membedakan konkurensi yang dibentuk oleh relasi AND dan OR.
7	*Time Based Discovery of Parallel Business Processes* (Riyanarto Sarno, Kartini, Widyasari Ayu Wibowo, Adhatus Solichah)	Metode ini merupakan modifikasi dari algoritma penemuan model proses berdasarkan *Activity Lifespan* (Riska A. Sutrisnowati, Hyerim Bae, Dongha Lee, Minsoo Kim). Dengan menggunakan *activity lifespan* dan prinsip *non-linear dependency*, metode ini dapat menemukan model proses. Dalam penelitian ini didefinisikan hubungan antar aktivitas di data log, yaitu *before and meets, overlaps, contains, is-finished-by, equals*, dan *starts*. Kemudian, dengan definisi ini, penulis mengusulkan teorema yang lebih kompleks yaitu hubungan *sequence* dan paralel. Kelebihan: Metode ini dapat menemukan model proses yang mengandung relasi *sequence*, paralel (AND, OR dan XOR), *noise*, *completeness*, dan penggunaan informasi *double timestamp*. Kekurangan: Metode ini tidak memberikan aturan atau syarat yang diperlukan untuk hubungan paralel model proses selain dengan menggunakan informasi *timestamp*.

Dengan mengacu pada berbagai penelitian yang melibatkan algoritma berbasis waktu sebagai algoritma dalam penemuan model proses, penelitian ini akan memodifikasi algoritma Alpha yang menggunakan prinsip hubungan resiprokal antar aktivitas menjadi berbasis informasi waktu ganda (*double timestamp*) agar dapat menemukan model proses yang mengandung hubungan *sequence*, paralel (AND, OR, XOR), *noise*, dan *completeness*. Algoritma ini juga akan dimodifikasi dalam kaitannya dengan hubungan paralel model proses dengan aturan tambahan atau syarat penting dari informasi *timestamp*.

2.1.2 Optimasi Waktu dan Biaya

Tabel 2.2. Penelitian terkait optimasi waktu dan biaya

Tahun	Area	Algoritma/ Metode	Masalah yang ditangani	Keterangan
2004	Manajemen	*Goal attainment method*	TCTP	Meminimalkan total biaya dan waktu rata-rata penyelesaian proyek
2007	Konstruksi	GA, *integer programming*	TCTP	Memaksimalkan keuntungan proyek

				melalui biaya langsung, biaya *overhead*, biaya pendanaan dan sumber daya yang berfluktuasi di bawah batas kredit dan sumber daya
2007	Manajemen dan Industri	MOGA, *fuzzy logic*	TCTP	Dapat memecahkan masalah dalam berbagai kondisi ketidakpastian yang ada dalam proyek realistis
2008	Manajemen dan Industri	GA	TCTP	Perbandingan antara algoritma klasik GA dan Siemens dilakukan dan ditemukan bahwa algoritma GA lebih sesuai untuk menangani masalah besar dan kompleks.
2011	Konstruksi	*Stochastic dominance rule*	TCQTP	Optimasi waktu, biaya dan kualitas dengan berbagai alokasi sumber daya
2011	Konstruksi	MILP. lingo 12	TCTP	Minimalkan total biaya proyek
2011	Industri	MOGA	TCTP	Penelitian mengembangkan pendekatan algoritma GA Pareto untuk menyelesaikan CPMTCTP dan menemukan solusi optimal yang berguna untuk proyek besar karena kecepatan tinggi dan konvergensi cepat.
2015	Industri dan IT	*Linear Programming*	TCTP	*Crashing* proyek dengan menggunakan CPM dan *linear programming* untuk memperoleh durasi minimum dan biaya tambahan.

Dengan mengacu pada berbagai penelitian yang melibatkan masalah *Time-cost trade off problem* (TCTP), penelitian ini menggunakan metode Sistem Manufaktur Fleksibel (FMS) untuk melakukan *scheduling* mesin dengan menggunakan FCFS (*First Come First Serve*) dan *Goal Programming* yang digunakan untuk mengoptimasi biaya dan waktu dari setiap aktivitas pada proses bisnis.

2.2 Dasar Teori

2.2.1 Model Proses Bisnis

Proses bisnis merupakan sekumpulan aktivitas yang dibuat untuk menghasilkan *output* spesifik dengan tujuan tertentu (Wang, He, Wen, Wu, ter Hofstede, & Su, 2010). Dari model proses tersebut juga dapat diketahui informasi di mana dan kapan suatu aktivitas dilakukan, kondisi awal sebelum aktivitas dilakukan, kondisi akhir setelah aktivitas dilakukan, serta *input* dan *output* yang jelas. Adapun ciri-ciri dari proses bisnis itu sendiri adalah sebagai berikut :

1. Mempunyai tujuan tertentu
2. Mempunyai *input* (masukan) yang spesifik.
3. Mempunyai *output* (keluaran) yang spesifik.
4. Memanfaatkan *resource*.
5. Memiliki aktivitas yang dapat dieksekusi dengan urutan tertentu.
6. Dapat melibatkan lebih dari satu organisasi.

Model proses bisnis merupakan representasi dari proses bisnis. Sehingga sebuah model proses bisnis harus secara jelas mendefinisikan setiap ciri-ciri yang harus dimiliki oleh suatu proses bisnis. *Unified Modeling Language* (UML) merupakan salah satu representasi dasar dari proses bisnis. Saat ini representasi dari model proses bisnis itu sendiri sudah banyak berkembang dan banyak jenisnya. Mulai dari *Causal Net*, UML, *Business* BPEL, *Business Process Model and Notation* (BPMN), EPC, PNML, dan masih banyak lagi. Tetapi, masing-masing jenis tersebut juga memiliki kegunaan dan fungsi sendiri-sendiri.

2.2.2 Data Log

Di dalam proses *mining*, untuk menganalisis suatu proses bisnis digunakan data log dari proses bisnis tersebut sebagai acuannya. Data log didefinisikan sebagai suatu set proses eksekusi yang mengambil data aktivitas proses bisnis yang dilakukan dalam konteks tertentu (Wen, Aalst, Jianmin, & Sun, 2012). Atau dengan kata lain, data log merupakan catatan dari eksekusi aktivitas dalam suatu proses bisnis. Catatan eksekusi ini dapat menyimpan data berupa waktu dilaksanakannya suatu aktivitas, *resource* yang melaksanakan aktivitas, dan lain-lain sesuai dengan kebutuhan dari perusahaan yang menjalankan data log-nya. Dalam data log dapat terdiri dari berbagai macam *case*, *trace*, dan *activity* (van der Aalst, Process Mining - Discovery, Conformance and Enhancement of Business Processes, 2011).

➨ *Case* dan *Trace*

Case merupakan suatu kasus tertentu yang ada pada data log. Kasus tersebut dapat berupa suatu kasus dalam memproduksi suatu barang tertentu, karena data log dapat terdiri dari catatan dari proses eksekusi pembuatan banyak barang atau proses eksekusi dari banyak kasus proses. Sedangkan *trace* merupakan alur dari aktivitas yang dijalankan dalam suatu proses. Misal dalam suatu data log *(E)*:

$$E = [< a, c, d >^{45}, < b, c, d >^{42}, < a, c, e >^{38}, < b, c, e >^{22}]$$

Dalam data log tersebut:

- Terdapat 4 *trace* yaitu (a,c,d), (b,c,d), (a,c,d), (b,c,e)
- Terdapat 147 *case* karena (a, c, d) dilakukan sebanyak 45 kali, (b, c, d) sebanyak 42 kali, (a, c, e) sebanyak 38 kali, dan (b, c, e) sebanyak 22 kali.

➨ Activity

Merupakan bagian dari *case* yang merupakan sub proses dalam pembuatan suatu barang atau dalam suatu proses tertentu. Misal pada data log *(E):*

$$L = [< a, c, d >^{45}, < b, c, d >^{42}, < a, c, e >^{38}, < b, c, e >^{22}]$$

Dalam data log tersebut terdapat 5 aktivitas yaitu {a, b, c, d, e}.

2.2.3 Process Mining

Process mining adalah penelitian baru berfokus antara *machine learning* dan *data mining* yang menangani pemodelan proses dan analisisnya. Tujuan *process mining* adalah menemukan, mengawasi, dan mengembangkan proses asli yang didapat dengan menggali informasi dari data log yang tersedia pada sistem saat ini (Aalst, 2010). Data log yang didapat dari sistem informasi saat ini sangat banyak dan sebagian besarnya tidak terstruktur. Sehingga pengekstrakan informasi dari sistem membutuhkan usaha yang lebih. *Process mining* terdiri dari 3 teknik yaitu, *discovery*, *conformance*, dan *enhancement.*

- *Discovery*

Salah satu teknik *process mining* yang bertujuan untuk mendapatkan model proses dengan menggali informasi dari data log. *Process discovery* terdiri dari beberapa algoritma seperti *alpha*, *alpha+*, *alpha++*, dan *genetic algorithm*.

- *Conformance*

Salah satu teknik *process mining* yang bertujuan membandingkan hasil model proses dari data log dengan model proses yang sudah ada. Kemudian, perbedaan diantaranya akan dianalisis lebih lanjut. Teknik ini dapat berguna dalam pengecekan kecurangan dalam proses bisnis (Riyanarno Sarno, Rahadian Dustrial Dewandono, Tohari Ahmad,

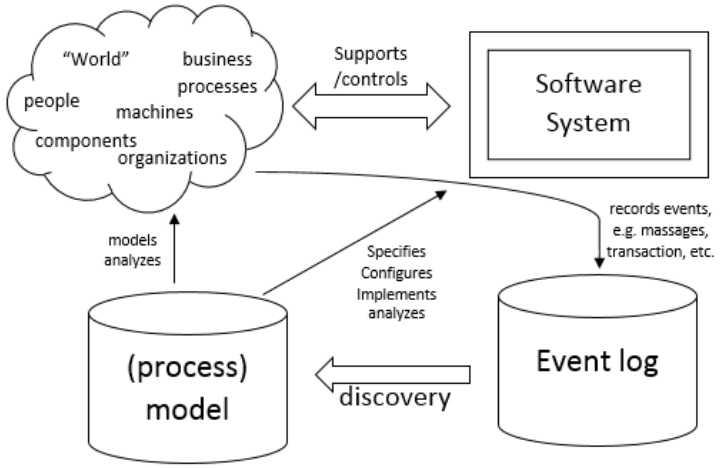
Mohammad Farid Naufal, dan Fernandes Sinaga, 2015) (Solichul Huda, Riyanarno Sarno, Tohari Ahmad, Heru Agus Santoso, 2014).

- *Enhancement*

Salah satu teknik *process mining* yang bertujuan untuk mengembangkan proses bisnis yang sudah ada. Model proses dari data log dianalisis untuk mendapatkan bagian proses yang masih dapat atau perlu dikembangkan.

2.2.4 Process Discovery

Process discovery merupakan salah satu proses yang paling menantang dari rangkaian *process mining*. Tujuan dari proses ini adalah untuk membentuk model dengan cara menggali informasi dari data yang tercatat dalam suatu data log (Goedertier, De Weerdt, Martens, Vanthienen, & Baesens, 2011). Dalam struktur proses bisnis, model dapat dianggap sebagai *graph* untuk mengandung satu set *node* dihubungkan dengan *edge* (Sarno, Ginardi, Pamungkas, & Sunaryono, 2013).



Gambar 2.1 Skema *Process Discovery*

Adapun algoritma yang digunakan dalam *Process Discovery* adalah algoritma *alpha miner, alpha plus miner, alpha plus plus miner, heuristics miner* (van der Aalst, Process Mining - Discovery, Conformance and Enhancement of Business Processes, 2011). Setiap algoritma memiliki pendekatan yang berbeda-beda dalam menganalisis proses yang terjadi. Dalam penelitian tesis ini akan mengembangkan algoritma Alpha berbasis waktu untuk melakukan *process discovery.*

2.2.5 Sistem Manufaktur Fleksibel (FMS)

Sistem Manufaktur Fleksibel (FMS) adalah satu atau lebih mesin produksi yang diintegrasikan dengan pemindahan material secara otomatis, dimana operasinya diatur dengan komputer. Definisi lain menyebutkan bahwa FMS merupakan sebuah sistem produksi yang terintegrasi dimana ada beberapa jumlah fleksibilitas yang memungkinkan

sistem untuk bereaksi dalam setiap perubahan, baik yang diperkirakan maupun yang tidak diperkirakan. Fleksibilitas dalam sistem manufaktur sering digambarkan sebagai:

1. Kemampuan untuk beradaptasi sesuai perubahan *engineering*
2. Peningkatan jumlah bagian yang sama yang diproduksi dalam suatu sistem
3. Kemampuan mengakomodasi perubahan rute yang memungkinkan sebagian dari produk diproduksi oleh mesin yang berbeda
4. Kemampuan untuk merubah *setup* sistem dengan cepat dari satu tipe produksi ke yang lainnya.

Adapun macam-macam fleksibilitas pada FMS adalah:

1. Fleksibilitas Mesin (*Machine Flexibility*)
2. Fleksibilitas Rute (*Routing Flexibility*)
3. Fleksibilitas Proses (*Process Flexibility*)
4. Fleksibilitas Produk (*Product Flexibility*)
5. Fleksibilitas Produksi (*Production Flexibility*)
6. Fleksibilitas Ekspansi (*Expantion Flexibility*)

Elemen kunci yang termasuk di dalam FMS adalah :

- Sebuah sistem penanganan material yang otomatis dan fleksibel dimana hal itu memungkinkan pekerja untuk beralih di antara setiap pasang mesin sehingga setiap *routing* pekerjaan dapat diikuti.
- Seperangkat stasiun kerja yang memuat peralatan mesin yang tidak memerlukan *set up time* yang signifikan atau mengubah urutan pekerjaan.
- Sebuah jaringan dan mikroprosessor pengawasan komputer yang melakukan beberapa tugas-tugas berikut :
    - mengarahkan *routing* pekerjaan melalui sistem.
    - melakukan pemeriksaan terhadap setiap status pekerjaan.
    - melewati setiap instruksi untuk setiap proses dan memastikan bahwa alat yang tersedia tepat untuk pekerjaan tersebut.
    - pemantauan kinerja operasi.

Dari uraian di atas, dapat kita lihat beberapa keuntungan dari konsep FMS, adalah:

- Mempermudah untuk menambah *line* produksi baru dan mengurangi kecelakaan kerja yang biasa terjadi pada *line* produksi.
- Mempermudah penanganan jika terjadi perubahan jumlah produksi, baik terjadi penambahan ataupun pengurangan kapasitas produksi.
- Perubahan desain dapat dilakukan dengan mudah dengan kontrol komputer.
- Meningkatkan efisiensi dalam penggunaan peralatan/mesin.
- Meningkatkan kualitas produk dan menjaga konsistensi kualitas produk.
- Mengurangi biaya ongkos pekerja (*men power*).

- Mengurangi luas lantai produksi (pada industri modern hal ini merupakan keuntungan yang dapat diperhitungkan).

Metode-metode yang dapat digunakan untuk menyelesaikan permasalahan pada Sistem Manufaktur Fleksibel (FMS) diantaranya:

- **FCFS (*FIRST COME FIRST SERVE*) / FIFO (*FIRST IN FIRST OUT*)**

FCFS/FIFO bisa diartikan sebagai proses yang tiba lebih dahulu akan dilayani lebih dahulu. Jika ada proses tiba pada waktu yang sama, maka pelayanan mereka dilaksanakan melalui urutan mereka dalam antrian. Proses di antrian belakang harus menunggu sampai semua proses di depannya selesai. Setiap proses yang berada pada status *ready* dimasukkan ke dalam FCFS *queue* sesuai dengan waktu kedatangannya.

Contoh Soal :

Jika diketahui terdapat 5 macam antrian proses, yaitu A-B-C-D-E dengan waktu kedatangan semuanya 0-1-2-2-5. Lama proses berturut-turut antara lain: 5-2-6-8-3.

Pertanyaan:
Kapan dimulainya eksekusi dari tiap-tiap antrian proses tersebut?

Kapan selesai eksekusinya?

Hitung *Turn Arround Time* (TA)-nya?

Berata rerata TA?

Rumus
TA = Waktu Tunggu + Lama Eksekusi

Rerata TA = ∑TA / ∑Job

Waktu Tunggu = Mulai Eksekusi – Waktu Tiba

Penyelesaian:

Proses (1)	**Waktu Tiba (2)**	**Lama eksekusi (3)**	**Mulai eksekusi (4)**	**Waktu Tunggu (5)**	**Selesai eksekusi (3)+(5)=(6)**	**TA (3) + (5) = (7)**
A	0	5	0	0	5	5
B	0	2	5	5	7	7
C	0	6	7	7	13	13
D	0	8	13	13	21	21
E	0	3	21	21	24	24

- **SJF (*Shortest-Job First*)**

Selain FCFS/FIFO ada juga yang namanya SJF (*shortest job first*) bisa diartikan yaitu setiap proses yang ada di *ready queue* akan dieksekusi berdasarkan *burst time* terkecil. Mengakibatkan *waiting time* yang pendek untuk setiap proses dan *waiting time* rata-ratanya juga menjadi pendek.

Langkahnya:

Langkah I: tentukan urutan prioritas berdasarkan pendeknya proses yang dilayani

Langkah II: penentuan proses mana yang dilayani oleh pemroses.

Contoh soal (dengan waktu tiba berbeda):

Proses	Lama Eksekusi	Waktu Tiba
D	1	0
E	3	2
B	5	5
C	7	7
A	10	9

Penyelesaian:

Proses	Waktu Tiba (1)	Lama eksekusi (2)	Mulai eksekusi (3)	Selesai eksekusi (4)	Waktu Tunggu (3)–(1)=(5)	TA (2) + (5) = (6)
D	0	1	0	1	0	1
E	2	3	2	5	0	3
B	5	5	5	10	0	5
C	7	7	10	17	3	10
A	9	10	17	27	8	18

2.2.6 Data Log dengan *Double Timestamp*

Saat ini, tidak semua aktivitas dalam proses bisnis dijalankan secara berurutan. Beberapa aktivitas dapat dilakukan bersamaan. Aktivitas yang dijalankan bersamaan adalah beberapa kegiatan yang berada pada beberapa kondisi, sebagai berikut:

1. Waktu mulai aktivitas berikutnya adalah sebelum waktu selesai aktivitas saat ini.
2. Waktu mulai aktivitas berikutnya dan aktivitas saat ini sama, namun aktivitas berikutnya selesai sebelum aktivitas saat ini selesai.

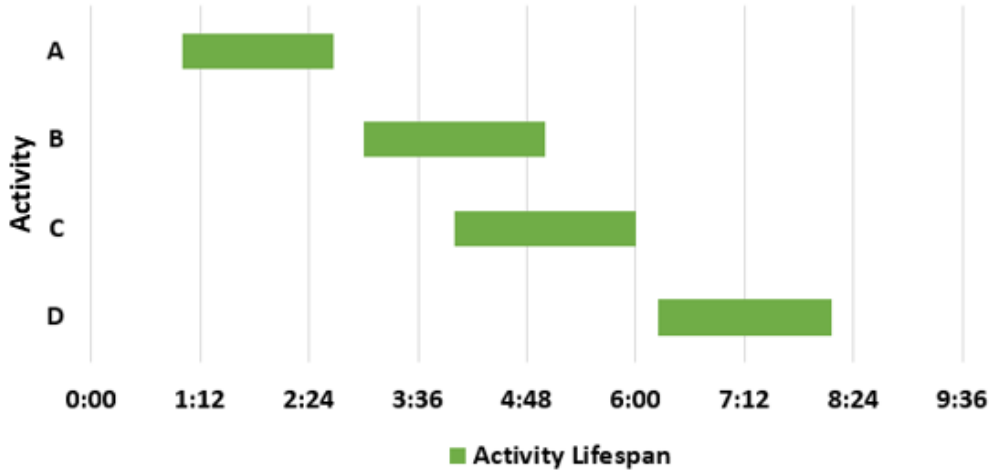
3. Aktivitas saat ini dan aktivitas berikutnya memiliki waktu mulai dan waktu selesai yang sama.

Manfaat utama dari aktivitas yang dilakukan bersamaan adalah meminimalkan waktu eksekusi suatu proses. Dengan menggunakan data log dengan *single timestamp*, aktivitas yang dilakukan bersamaan sulit dideteksi. Hal ini karena data log dengan *single timestamp* hanya mencatat waktu mulai aktivitas, sedangkan untuk mengetahui aktivitas yang dilakukan bersamaan membutuhkan waktu mulai dan selesai. Ini adalah salah satu alasan data log dengan d*ouble timestamp* terjadi.

Tabel 2.3 menunjukkan contoh dari data log dengan *double timestamp*. Aktivitas yang dilakukan bersamaan di dalam data log adalah aktivitas '*Getting Reviews from Judges C*' dan aktivitas '*Getting Review of Judges A*'. Hal ini sesuai dengan kondisi pertama dari aktivitas yang dilakukan bersamaan, yaitu waktu mulai aktivitas berikutnya adalah sebelum waktu selesai aktivitas saat ini. Waktu mulai aktivitas '*Getting Review of Judges A*' adalah sebelum waktu selesai dari aktivitas '*Getting Review of Judges C*'. Beberapa algoritma telah menggunakan data log dengan *double timestamp*, seperti algoritma Heuristics Miner dengan interval waktu.

Tabel 2.3. Contoh dari data log dengan *double timestamp*

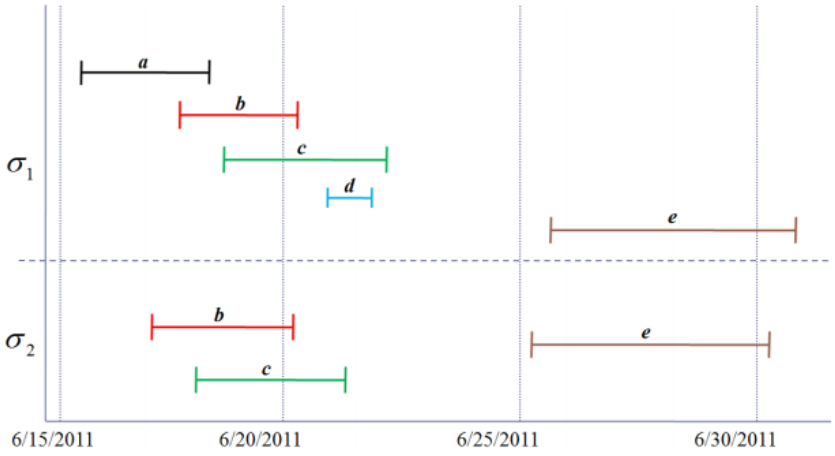
Case ID	Activity	Start Stamp	Finish Stamp	Resource
1	Inviting Reviewers	3/9/2016 11:24	3/9/2016 11:27	John
1	Getting Reviews from Judges	3/9/2016 11:27	3/9/2016 11:30	Leo
1	Getting Reviews from Judges C	3/9/2016 11:30	3/9/2016 11:33	Layla
1	Getting Reviews from Judges A	3/9/2016 11:32	3/9/2016 11:35	Leo
1	Collecting Reviews	3/9/2016 11:35	3/9/2016 11:38	Smith
1	Checking Reviews from Judges	3/9/2016 11:38	3/9/2016 11:41	Carlotte
1	Deciding Final Result of Journal based on Reviews	3/9/2016 11:41	3/9/2016 11:44	Novi, Fiona, Helli
1	Determining The Final Result as Accepted Journal	3/9/2016 11:44	3/9/2016 11:47	Novi, Fiona, Helli
1	Annoucing the Final Result	3/9/2016 11:47	3/9/2016 11:50	John



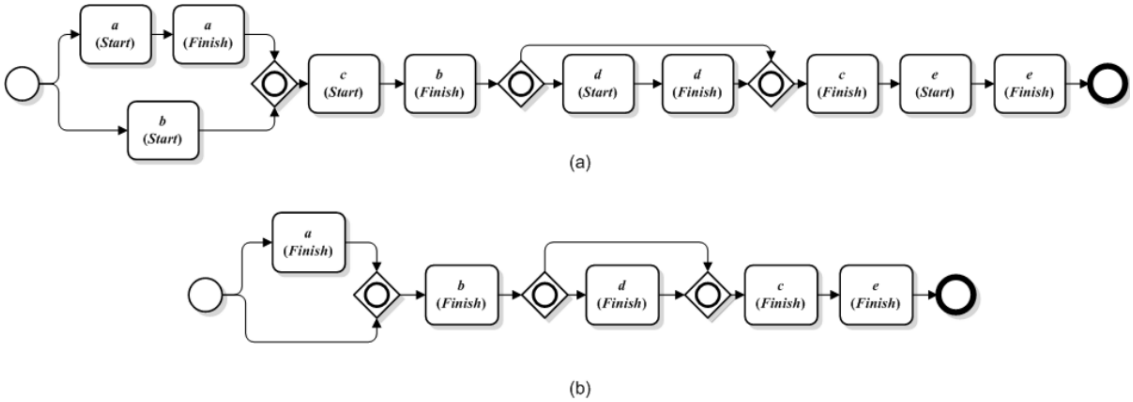
Gambar 2.2 Data log dengan *double timestamp* pada *activity lifespan*

2.2.7 *Single timestamp* dan *Double timestamp*

Single timestamp adalah waktu mulai atau selesai sebuah aktivitas dalam proses bisnis yang dicatat dalam data log. Sedangkan *double timestamp* adalah waktu mulai dan selesai sebuah aktivitas dalam proses bisnis yang dicatat dalam data log. Selisih waktu akhir dan mulai sebuah aktivitas biasa disebut *activity lifespan* atau durasi dalam *process discovery. Activity lifespan* dapat dilihat dalam *double timestamps* data log seperti pada Gambar 2.2. *Process discovery* yang menggunakan *double timestamp* memiliki keunggulan dalam *incompleteness* daripada *single timestamp*. Hal ini disebabkan *single timestamp* membutuhkan *trace* dua kali lebih banyak daripada *double timestamp* dalam kasus model yang mengandung paralel. Dalam kasus tersebut, *single timestamp* membutuhkan hubungan resiprokal dalam menggali relasi paralel sehingga menggunakan dua *trace* dalam menggali paralel antara aktivitas A dan B (AB, BA) (Aalst, 2010). Sedangkan *double timestamp* menggunakan *activity lifespan* untuk menggali relasi paralel sehingga cukup menggunakan satu *trace* saja dalam menggali paralel antara aktivitas A dan B (AB). Berikut adalah contoh data log dalam bentuk *Gantt chart* pada Gambar 2.3 (Riska A. Sutrisnowati, Hyerim Bae, Dongha Lee, Minsoo Kim, 2014).



Gambar 2.3 Data log pada *Gantt Chart*



Gambar 2.4 (a) Model proses menggunakan *double timestamp;* (b) Model proses menggunakan *single timestamp*

Model proses yang didapat dari data log pada Gambar 2.3 digambarkan pada Gambar 2.4. Gambar 2.4a adalah gambar model proses menggunakan *double timestamp*. Penggalian informasi dari data log menggunakan waktu mulai dan selesainya aktivitas. Gambar 2.4b adalah gambar model proses menggunakan *single timestamp*. Penggalian informasi dari data log menggunakan waktu selesainya aktivitas saja. Tabel 2.4 menjelaskan perbedaan *double timestamp* dan *single timestamp.*

Tabel 2.4. Perbedaan antara *single timestamp* dan *double timestamp*

Single timestamp	***Double timestamp***
waktu mulai atau selesai sebuah aktivitas dalam proses bisnis yang dicatat dalam data log	waktu mulai dan selesai sebuah aktivitas dalam proses bisnis yang dicatat dalam data log
single timestamp membutuhkan *trace* dua kali lebih banyak daripada *double timestamp* dalam kasus model yang mengandung paralel	

single timestamp membutuhkan hubungan resiprokal dalam menggali relasi paralel	*double timestamp* menggunakan *activity lifespan* untuk menggali relasi paralel

2.2.8 Completeness

Completeness adalah suatu kondisi dimana data log menyimpan seluruh perilaku yang bisa dieksekusi pada proses bisnis (Agrawal R., Gunopulos D., dan Leyman F., 1998). Process discovery memiliki gagasan *completeness* masing-masing. *Process discovery* bekerja dengan benar apabila data log yang digali memenuhi gagasan completeness algoritma tersebut. Data log yang tidak memenuhi gagasan *completeness* disebut *incompleteness*. *Incompleteness* terjadi karena terlalu sedikit data pada data log yang mengakibatkan kesalahpahaman pada saat *process discovery* dilakukan. Sehingga model proses yang dihasilkan tidak seperti yang seharusnya.

Secara umum, sebuah model dengan 10 aktivitas paralel membutuhkan 10! = 3,628,800 *trace*. Dengan jumlah yang begitu besar, data log tidak mungkin mencatat seluruh *trace* tersebut. Selain itu, tiap *trace* tidak memiliki kesempatan yang sama untuk dieksekusi sehingga *trace* yang kesempatannya sedikit memiliki kemungkinan yang lebih kecil untuk disimpan pada data log. Oleh karena itu, dapat dikatakan bahwa data log tidak realistis untuk komplit. Sehingga algoritma *process discovery* harus menemukan cara untuk mengurangi jumlah *trace* dalam gagasan *completeness*nya (Aalst, 2010).

Algoritma alpha memiliki gagasan *completeness* yaitu n(n-1). Apabila sebuah model dengan 10 aktivitas paralel, maka data log yang dapat digunakan algoritma alpha adalah data log yang memiliki 10(10-1) = 90 *trace*. Jumlah ini sangat realistis dibanding 10!.

Tabel 2.5. Penelitian terkait *Completeness*

Penulis dan Judul Penelitian	**Algoritma/ Metode**	**Kelebihan**	**Kekurangan**
Mining Process Models with Prime Invisible Tasks (Lijie Wen, dkk)	Algoritma *Alpha++*	Penjabaran algoritma nya mudah dipahami	Model yang mining tidak *soundness* karena *deadlock*. Algoritma hanya dapat digunakan pada WF net, bukan data log
Process Mining - Overview and Opportunities (van der aalst)	Algoritma *Alpha*	*Discover concurrency* *Discover invisible* dalam data log	Belum dapat menangani *noise*, *loop* dan *non-local dependency*

			Penjelasan dari *completeness* belum jelas
CoBeFra for Conformance Analysis between Procedural Process Models and Event Logs in ProM (Broucke, dkk)	*Comprehensive benchmarking framework* (CoBeFra)	Waktu proses berjalan cepat *Fault tolerant* selama proses eksekusi berlangsung	Hanya dapat melakukan pengecekan/analisis model proses hasil *discovery* metode lain
Discovering Block-Structured Process Models from Event Logs Containing Infrequent Behaviour (Leemans, dkk)	*Divide and conquer*	*Discover soundness* dari model proses dengan cepat	Hasil *fitness* yang rendah
Process Discovery and Conformance Checking Using Passages (Aalst, verbeek)	*Divide and conquer*	Tingkat pemrosesan yang lebih cepat Efisien	*Trade-off speed* (kecepatan) dan *passage* Kemungkinan *merging* yang tinggi Tidak ada metode yang dapat mengatasi *incompleteness* di data log
Predictive Monitoring of Business Processes (Maggi, dkk)	Pendekatan *Predictive Monitoring* (*Decision Tree*)	Menghindari prediksi	Menunjukkan akurasi yang rendah Klasifikasi *decision tree* lemah dalam menangani data set yang besar
An effective parallel approach for genetic-fuzzy data mining (Hong, dkk)	*Genetic – Fuzzy mining*	Efisien dan pemrosesan yang cepat Evaluasi fitness value tinggi	Tidak efisien jika jumlah populasi besar Metode yang diusulkan hanya dapat diterapkan pada arsitektur *master-sleeve*
Review of Web Structure Mining	*Clustering and Ranking*	Metode yang diusulkan dapat	Terdapat masalah dalam penemuan

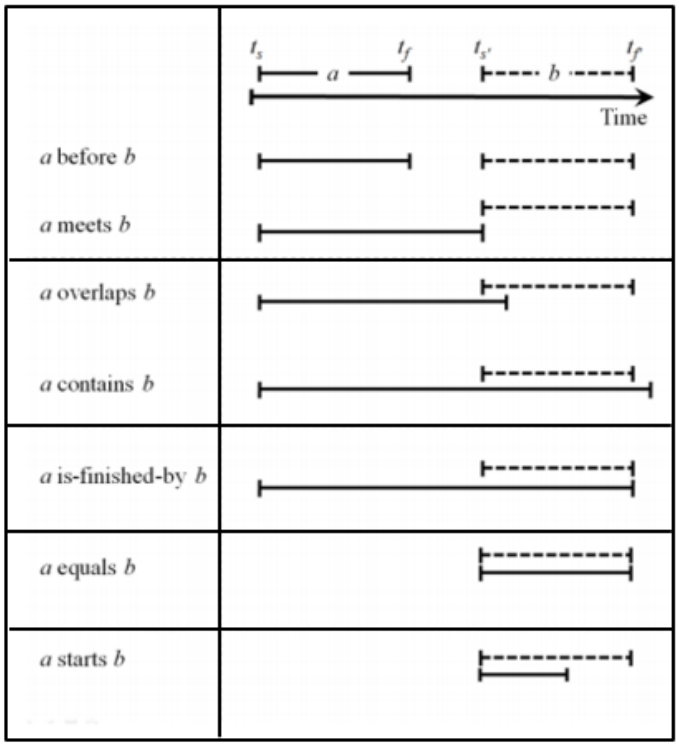
Techniques using Clustering and Ranking Algorithms (Sharma, Kaur)		meningkatkan efisiensi Meningkatkan *execution time*	*clusters* yang berisi dokumen yang relevan
A comparative study of fuzzy logic and weighted Association rule mining in frequent datasets (Rubia, dkk)	*Weighted Association Rule Mining*	Penggunaan memori yang sedikit Efisiensi dalam menghasilkan *association rule* dengan menggunakan manipulasi matriks	Terdapat masalah dalam menghasilkan *association rule* dari data yang jumlahnya besar
Merging Event Logs for Process Mining (Jan Claes, Poels)	*Rule Based Merging* dan *Rule Suggestion Method*	Penggabungan data log dari *partner* berbeda dari inter-organisasi dijelaskan dengan konsep yang baik	Tidak membahas mengenai *incompleteness* dalam data log
Aligning Real Process Executions and Prescriptive Process Models through Automated Planning (de Leoni, dkk)	*Alignment method*	Metode yang diusulkan dapat memperbaiki data log yang diujikan dalam studi kasus	Metode ini membutuhkan model proses yang telah dideklarasikan sebelumnya untuk proses alignment
Model repair - aligning process models to reality (Fahland, Aalst)	*Alignment*	Metode yang diusulkan dapat menemukan *loop* dalam model proses	Tidak dapat mengatasi masalah generalisasi dan presisi Masalah dalam penemuan titik *alignment* yang tepat/sesuai
A framework for efficiently mining the organisational perspective of business processes (Schonig, dkk)	*Resource-aware and declarative process mining*	Metode yang diusulkan dapat meningkatkan efisiensi	*Dependency* antar *case* tidak dapat di-*discover*

		Model proses dalam bentuk yang sederhana	
On the gap between reality and registration - a business event analysis classification framework (Broucke, dkk)	*Business process analytics framework*	Memberikan skema penamaan yang jelas untuk proses bisnis	Mengasumsikan data log *completeness*
Event log imperfection patterns for process mining towards a systematic approach to cleaning event logs (Suriadi, dkk)	*Patterns-based approach*	Metode yang diusulkan dapat meningkatkan kualitas dari data log dan model proses	Tidak ada *framework* formal untuk menentukan dan menggambarkan pola
Applying Process Mining Techniques in Software Process Appraisals (do Valle)	*Extended process mining method*	Metode yang diusulkan dapat meningkatkan limitasi dari data log	Tidak ada standar dari metrik evaluasi

2.2.9 *Temporal Causal Relation*, *Control Flow*, dan *Double Timestamp* dalam Instans Proses

Definisi 2.1 *Temporal Causal Relation*.
Relasi *causal* ini dideskripsikan pada Gambar 2.5. Relasi ini menunjukkan relasi *concurrent* antara dua aktivitas *A* (A_s, A_f) dan *B* (B_s, B_f) dimana A_s sebagai waktu mulai eksekusi aktivitas A dan A_f sebagai waktu selesai eksekusi aktivitas A, serta terdapat *data log* (*L*) dan *trace* (σ) sehingga $\sigma \in L$ dan $A, B \in L$ dapat dibedakan sebagai berikut:

$$Before\ and\ meets, A > B\ iff\ A_f \leq B_s$$
$$Overlaps, A \square B\ \ iff\ A_f > B_s\ \wedge\ A_f < B_f$$
$$Contains, A@B\ iff\ A_s < B_s\ \wedge\ B_f > A_s\ \wedge\ \ A_f > B_f$$
$$Is\ finish\ by, A_f B\ iff\ A_f = B_f \wedge\ A_s < B_s\ \wedge\ B_s < A_f$$
$$Equals, A \Diamond B\ iff\ A_s = B_s\ \wedge\ A_f = B_f$$
$$Starts, A_p B\ iff\ A_s = B_s\ \wedge\ A_f > B_f$$



Gambar 2.5 ***Temporal Causal Relation***

Definisi 2.2 *Control Flow*.
Relasi *control flow* antara dua aktivitas *A* (A_s, A_f) dan *B* (B_s, B_f) dimana terdapat *data log* (*L*) dan *trace* (σ) sehingga $\sigma \in L$ dan *A*, $B \in L$ dapat dibedakan sebagai berikut:

$$Sequence, A \rightarrow B \ \ iff \ A > B$$

$$Parallel, A \parallel B \ \ iff \ A > B \ \wedge B > A \ \vee \{A \square B \ \vee \ A@B \ \vee \ A_f B \ \vee A \Diamond B \ \vee A_p B\}$$

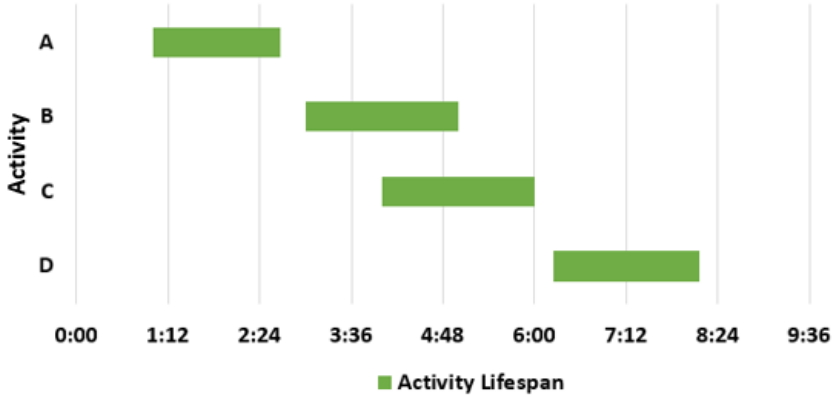
Definisi 2.3 *Double timestamp* dalam Instans Proses.
Diberikan *data log L* dan *trace* σ sehingga $\sigma \in L$. Relasi *sequence* A→B dan A→C antaraktivitas $A(e_s,e_f)$, $B(e_s,e_f)$, dan $C(e_s,e_f)$ menyebabkan A>B, A>C, dan B||C. Begitu juga relasi *sequence* B→D dan C→D antaraktivitas $A(e_s,e_f)$, $B(e_s,e_f)$, dan $C(e_s,e_f)$ menyebabkan B>D, C>D, dan B||C.

2.2.10 Relasi Model Proses Bisnis

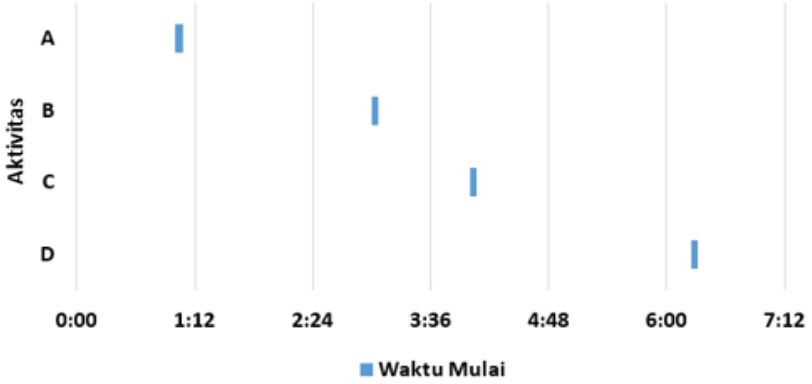
Relasi antaraktivitas dalam model proses bisnis terdiri dari relasi *sequence* dan paralel. Relasi *sequence* adalah relasi yang mengindikasikan eksekusi secara berurutan antaraktivitas yang berelasi *sequence*. Relasi paralel adalah relasi yang mengindikasikan eksekusi secara bersamaan atau berurutan antaraktivitas yang berelasi paralel. Relasi paralel terdiri dari paralel AND, *conditional OR,* dan *conditional XOR.* Eksekusi antaraktivitas paralel secara bersamaan atau berurutan ini hanya dapat diketahui apabila *process discovery* yang digunakan menggunakan *double timestamp.* Hal ini disebabkan,

double timestamp menggunakan data waktu mulai dan akhirnya eksekusi tiap aktivitas seperti yang ditunjukkan pada Gambar 2.6 sehingga dapat melihat apakah antaraktivitas dieksekusi secara bersamaan atau berurutan. Sedangkan *single timestamp* hanya melihat eksekusi antaraktivitas secara berurutan saja seperti pada Gambar 2.7.

Relasi-relasi ini digali dari data log untuk membangun suatu model proses bisnis. Namun, karena eksekusi relasi paralel dapat dilakukan secara bersamaan atau berurutan, relasi antaraktivitas ini bisa dicatat sebagai relasi *sequence* atau *concurrent* dalam data log. Relasi *concurrent* pada Gambar 2.8 adalah relasi yang mengindikasikan eksekusi secara bersamaan antaraktivitas yang berelasi *concurrent*.

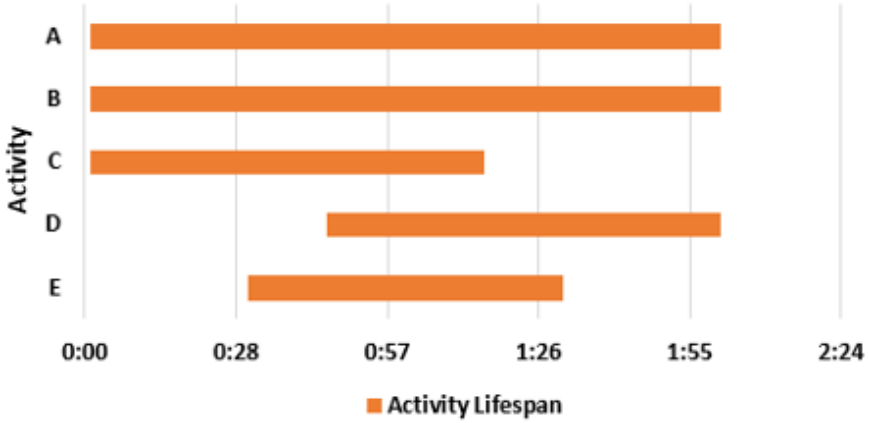


Gambar 2.6 *Double timestamp* data log



Gambar 2.7 *Single timestamp* data log

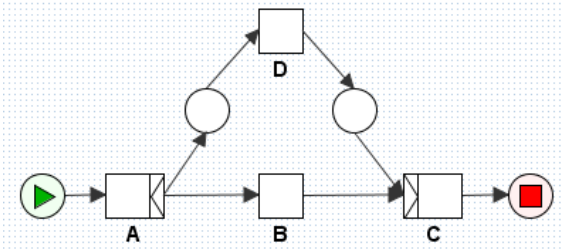
Relasi antaraktivitas ini menentukan relasi masukan dan keluaran suatu aktivitas dalam model proses bisnis. Relasi *sequence* menyebabkan aktivitas dengan relasi tersebut hanya menuju atau dituju satu aktivitas saja. Sedangkan akibat relasi paralel antaraktivitas, terdapat aktivitas yang menuju ke lebih dari satu aktivitas. Aktivitas yang dituju lebih dari satu aktivitas memiliki relasi masukan "*join"*. Dan Aktivitas yang menuju lebih dari satu aktivitas memiliki relasi keluaran "*split"*.



Gambar 2.8 Relasi *concurrent* pada data log

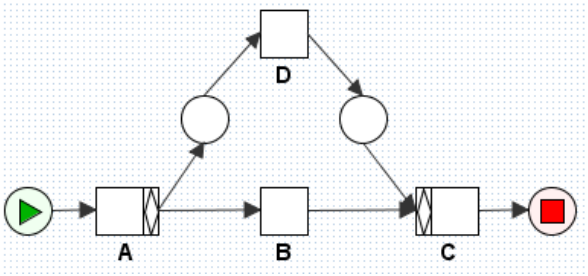
Relasi *join* adalah relasi antara beberapa aktivitas (*predessor*) menuju satu aktivitas (*successor*). Relasi *split* adalah relasi antara satu aktivitas (*predessor*) menuju beberapa aktivitas (*successor*). *Predessor* adalah aktivitas yang berjalan saat ini. *Successor* adalah aktivitas yang selanjutnya akan dijalankan setelah *predessor*. Relasi *split* dan *join* ini dibedakan berdasarkan relasi paralel yaitu paralel AND *split*, *conditional OR split*, *conditional XOR split*, paralel AND *join*, *conditional OR join*, dan *conditional XOR join*.

- Paralel AND
  Relasi paralel AND *split* terjadi ketika *predessor* dapat berjalan bersamaan. Namun, tidak ada aktivitas yang tidak berjalan ketika ada aktivitas *predessor* diantaranya yang berjalan. Relasi parallel AND *join* terjadi ketika *successor* harus menerima keluaran dari seluruh aktivitas *predessor*nya.



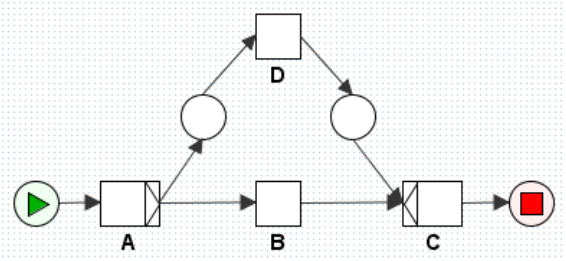
Gambar 2.9 Paralel AND

- *Conditional OR*
  Relasi *conditional OR split* terjadi ketika *predessor* dapat berjalan bersamaan. Berbeda dengan paralel AND, *conditional OR* memilih aktivitas yang akan dijalankan apabila memenuhi kondisi. Sehingga ada aktivitas yang bisa tidak dijalankan diantara aktivitas *successor*nya. Relasi *conditional OR join* terjadi ketika *successor* bisa menerima keluaran dari satu atau lebih aktivitas *predessor*nya.



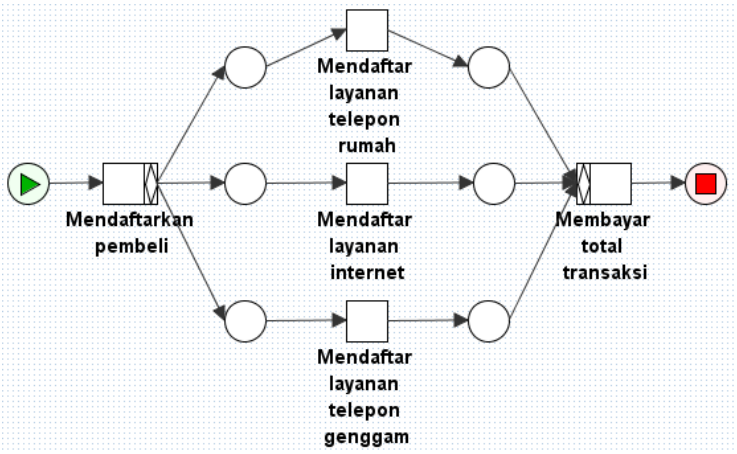
Gambar 2.10 *Conditional OR*

- *Conditional XOR*
  Relasi *conditional XOR split* terjadi ketika hanya satu dari seluruh *predessor* dapat berjalan. Relasi *conditional XOR join* terjadi ketika *successor* hanya menerima keluaran dari satu aktivitas *predessor*nya.



Gambar 2.11 *Conditional XOR*

Perbedaan antarparalel ini dijelaskan menggunakan model proses pada Gambar 2.12. Model proses tersebut menggambarkan proses bisnis kegiatan pendaftaran layanan berbayar yang terdiri dari layanan telepon rumah, internet, dan telepon genggam. Berdasarkan proses bisnis ini, pengguna dapat mendaftar seluruh atau sebagian layanan saja karena relasi antara aktivitas "Mendaftar layanan telepon rumah", "Mendaftar layanan internet", dan "Mendaftar layanan telepon genggam" berupa *conditional OR*. Namun apabila berupa paralel AND, maka pembeli harus mendaftar seluruh layanan. Dengan kata lain, pembeli dipaksa mendaftar seluruh layanan walaupun layanan tersebut tidak saling bergantungan. Dan apabila *conditional XOR*, pembeli hanya diperbolehkan mendaftar salah satu layanan tiap transaksi. Dengan kata lain, pembeli yang ingin mendaftar lebih dari satu layanan harus mendaftar berulang kali.



Gambar 2.12 Model Proses Pendaftaran Layanan

2.2.11 Algoritma *Alpha Miner*

Langkah-langkah yang digunakan dalam Algoritma *Alpha* untuk menemukan *Workflow Net / Petri Net* dari *case* yang sudah diberikan adalah sebagai berikut:

1. Membuat sekumpulan transisi dari data log pada *Workflow net*.
2. Membuat sekumpulan transisi keluaran dari *source place* pada *Workflow net*.
3. Membuat sekumpulan transisi masukan dari *sink place* pada *Workflow net*.
4. Pada langkah ke-4 dan 5, digunakan untuk menentukan *place* dari *Workflow net* yang sudah ditemukan. Pada langkah ke-4 algoritma Alpha menentukan transisi mana yang berelasi secara *causal*, sehingga untuk setiap *tuple* (A, B) setiap transisi di set A akan berelasi secara *causal* pada semua transisi di set B dan tidak ada relasi di antara A dan B yang saling mengikuti (*follow*) satu sama lain.
5. *Set* hasil dari langkah-4 diperbaiki dengan hanya mengambil elemen yang paling besar. Langkah ke-5 menentukan jumlah pasti *places* yang dimiliki oleh *Workflow net*.
6. *Places* yang sebelumnya diidentifikasi dibuat.
7. Masing-masing *place* dihubungkan dengan transisi masukan/keluaran yang bersesuaian.
8. Algoritma kemudian mengembalikan *Workflow net* yang diperoleh dari langkah-langkah sebelumnya.

2.2.12 Goal Programming

Goal programming merupakan perluasan dari model *linier programming*. *Linier programming* merupakan suatu cara untuk meyelesaikan persoalan pengalokasian sumber- sumber yang terbatas seperti tenaga kerja, bahan baku, jam kerja mesin dan sebaginya dengan cara terbaik yang mungkin dilakukan sehingga diperoleh maksimasi yang dapat berupa maksimasi keuntungan atau maksimasi yang dapat berupa minimasi

biaya. (Tjuju, 2002). Cara terbaik yang dimaksud adalah keputusan yang diambil berdasarkan pilihan dari berbagai alternatif.

- Digunakan pada kondisi yang diinginkan yaitu minimasi dari tujuan/sasaran yang diharapkan
- *Goal Programming* adalah programa linier yang memiliki satu atau lebih *goal* sasaran yang memenuhi syarat linieritas, maka dapat diselesaikan sebagai programa linier yang ada (simpleks, dua fasa, atau primal dual)
- Minimasi simpangan (atas atau bawah, kiri atau kanan) dimaksudkan agar sasaran yang diinginkan memang dapat tercapai.
- Formulasi *goal programming* sedikit berbeda dengan programa linier (PL) biasa, yaitu
    - Fungsi tujuan berupa minimasi simpangan (deviasi) bawah.
    - Fungsi pembatas ditambah dengan pembatas dari sasaran yang diinginkan.
    - Penyelesaian dapat diselesaikan dengan metode dua fasa, walaupun hanya melibatkan 2 variabel keputusan.

Jenis *Goal Programming*

- *Goal Programming* yang mempunyai tujuan tanpa Prioritas
    - Semua *goals* dianggap setara (sama penting)
- *Goal Programming* yang mempunyai tujuan dengan Prioritas
    - Setiap *goal* memiliki tingkat urgensi yang berbeda
    - Prioritas 1 >>> Prioritas 2 >>> … Prioritas *n*

Formulasi Model *Goal Programming*

Formulasi Model *Goal Programming* permasalahan yang akan diselesaikan adalah penentuan kombinasi produk yang optimal. Dengan demikian, yang menjadi variabel keputusan adalah jumlah masing-masing jenis produk yang akan dibuat, misalnya: X_1, X_2, X_3 dan X_4. Adapun tujuan-tujuan yang ingin dicapai secara berurutan adalah meminimalkan biaya produksi dan meminimalkan waktu produksi.

Sasaran meminimalkan biaya produksi

Fungsi tujuan :

$$Min\ Z = \sum_{i=1}^{m} H_i X_i$$

dimana :

H_i = biaya produksi per unit produk i

X_i = jumlah produk i yang diproduksi

m = banyaknya jenis produk

Sasaran meminimalkan waktu produksi

Kendala sasaran :

$$d^+_j \leq JL_j$$

dimana :

JL_j = kapasitas maksimum jam kerja aktivitas/fasilitas j

Fungsi tujuan:

$$Min\ Z = \sum d^+_j$$

dimana :

d_1- = *underachievement* (penyimpangan dibawah) target profit

d_1+ = *overachievement* (penyimpangan diatas) target profit

(*halaman ini sengaja dikosongkan*)

BAB III
METODA PENELITIAN

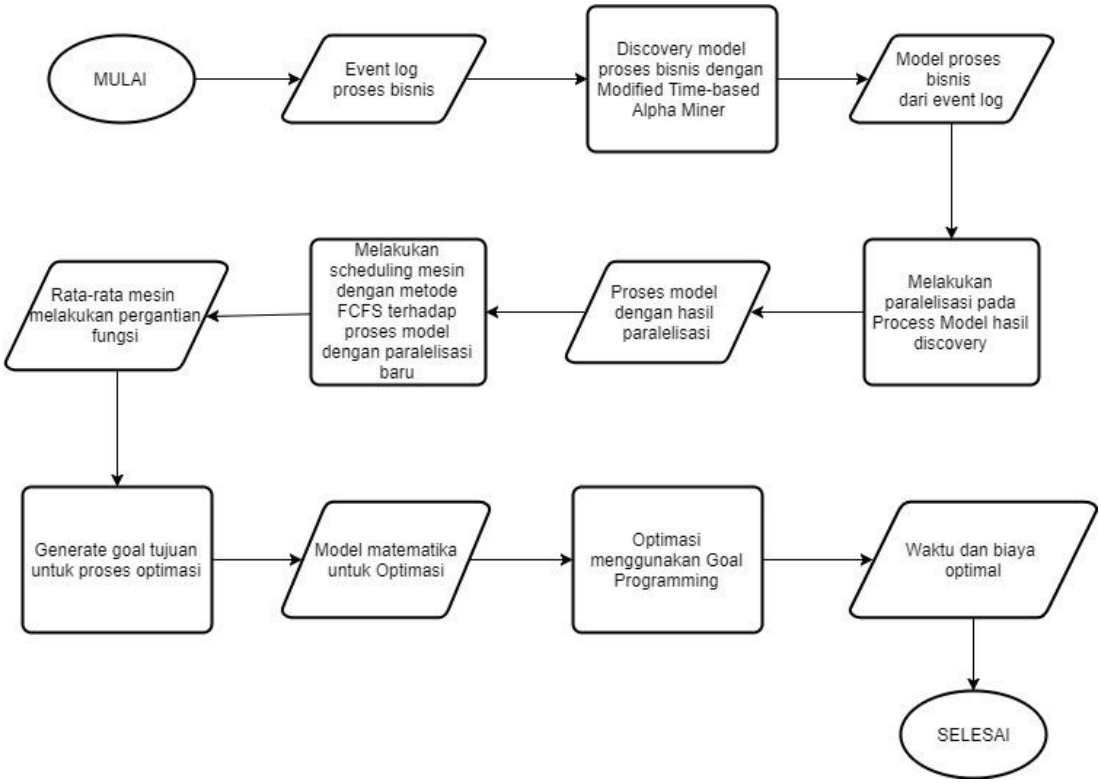
Bab ini akan memaparkan tentang metodologi penelitian yang terdiri dari metode usulan, *input*, *pre-processing*, *process discovery* dengan *Modified Time-based Alpha Miner*, kriteria untuk paralelisasi, *scheduling* dengan Sistem Manufaktur Fleksibel, optimasi waktu dan biaya dengan *Goal Programming*, dan *output*.

3.1 Metode Usulan

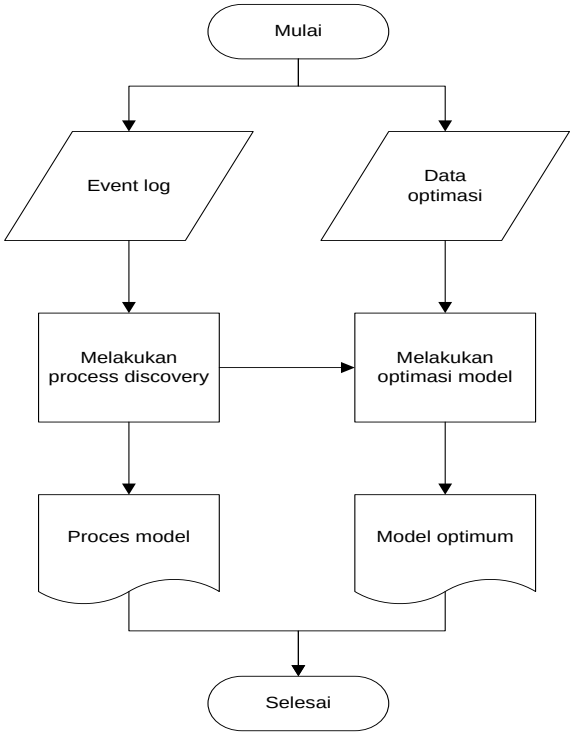
3.1.1 Model Sistem Metode Usulan

Metode usulan pada Tesis ini adalah menemukan model proses dari data log dengan modifikasi algoritma *Alpha Miner* yang awalnya hanya memperhatikan urutan antar aktivitas secara *sequence* menjadi berbasis interval waktu. Setelah model proses ditemukan, maka selanjutnya mencari kemungkinan aktivitas atau departemen yang dapat dieksekusi secara paralel untuk mempercepat waktu eksekusi. Dengan mengetahui aktivitas atau departemen mana dari model proses yang dapat dieksekusi secara paralel, maka akan muncul permasalahan baru yaitu penggunaan mesin sebagai *resource* pelaksana aktivitas akan mengalami kendala seperti *bottleneck* yang dapat berakibat keterlambatan selesainya proses suatu aktivitas. Oleh karena itu, untuk menyelesaikan permasalahan mesin tersebut, dilakukan scheduling dengan Sistem Manufaktur Fleksibel (FMS) dengan metode FCFS dan kakas bantu LEKIN untuk mengetahui departemen dan aktivitas apa yang terkena dampak pada saat mesin terlambat menyelesaikan jobnya. Hasil dari scheduling ini adalah rata-rata mesin dapat berganti fungsi setiap hari dan setiap bulan agar job mesin dapat terlaksana dengan baik tanpa adanya keterlambatan penyelesaian job. Setelah mesin di *schedule*, maka selanjutnya dilakukan optimasi waktu dan biaya dari model proses dengan *Goal Programming* dan kakas bantu Lingo. Tujuan optimasi adalah meminimalkan waktu terlama dari setiap departemen sehingga dapat menyesuaikan dengan rata-rata waktu eksekusinya. Jika waktu optimal diperoleh, maka diperoleh pula biaya yang dapat dihemat jika waktunya sudah diminimalkan.

Ada 7 tahapan utama dalam penelitian ini, yaitu *input*, *pre-processing*, *process discovery* dengan *Modified Time-based Alpha Miner*, kriteria untuk paralelisasi, *scheduling* dengan Sistem Manufaktur Fleksibel, optimasi waktu dan biaya dengan *Goal Programming*, dan *output*. Gambar 3.1 merupakan model metode usulan tesis secara detail dan Gambar 3.2 menjelaskan alur sistem yang dibangun pada penelitian ini.



Gambar 3.1 Metode usulan tesis



Gambar 3.2 Alur sistem

3.1.2 Jadwal Kegiatan

Jadwal kegiatan dalam penyusunan penelitian ini adalah sebagai berikut :

Tabel 3.1 Jadwal Kegiatan Penelitian

No.	Kegiatan	Bulan																			
		Agustus 2017				September 2017				Oktober 2017				November 2017					Desember 2017		
1.	Studi Literatur																				
2.	Analisa dan Perancangan																				
3.	Implementasi																				
4.	Pengujian dan Evaluasi																				
5.	Penyusunan Buku Tesis																				

3.2 Input

Dalam penelitian ini data yang didapatkan berasal dari PT. Terminal Petikemas Surabaya (PT TPS). Data yang diberikan untuk penelitian ini adalah data dari bulan Januari hingga Maret 2016 yang berjumlah 72.732 container pada kegiatan impor. *Format* data yang kami terima dari PT TPS adalah dalam *format database*. Departemen yang terlibat pada kegiatan impor ini adalah Departemen TPS, *Customer*, *Behandle*, dan Karantina. Untuk Departemen TPS menangani bagian proses *discharge* dan *delivery*. Departemen *Behandle* menangani 2 jalur *container*, yaitu jalur hijau (container yang masuk tidak diperiksa) dan jalur merah (container yang masuk diperiksa terlebih dahulu). Sedangkan Departemen Karantina menangani masalah pemeriksaan tumbuhan atau hewan dari luar negeri. Data awal dari PT TPS ditunjukkan pada Tabel 3.2.

Tabel 3.2 Data awal PT TPS

CONTA	CTR_S	CTR_T	GROSS	VESSEL_ATB	DISC_DAT	YARD_	YARD_	STACK_DATE	CUSTOMS_D	HAS_C	BEHAN	FIRST_STA	QUARANTIN	JOB_DEL_DA	TRUCK_IN_DA	TRUCK_OUT_D
4484646	40	DRY	29.103	1/31/16 14.35	2/1/16 20.22	M	112	2/1/16 21.00	2/4/16 0.00	YES			2/5/16 10.28	2/4/16 16.29	2/5/16 15.14	2/5/16 16.32
4484654	40	DRY	28.408	1/31/16 14.35	2/1/16 20.24	I	126	2/1/16 21.14	2/2/16 0.00					2/2/16 11.14	2/4/16 17.46	2/4/16 18.11
4485330	20	DRY	24.82	1/31/16 14.35	2/1/16 21.52	S	13	2/1/16 22.13	2/4/16 0.00					2/4/16 16.55	2/6/16 3.03	2/6/16 3.27
4484680	20	DRY	22.235	1/31/16 14.35	2/1/16 21.54	S	13	2/1/16 22.17	2/4/16 0.00	YES			2/5/16 16.11	2/4/16 15.06	2/9/16 17.35	2/9/16 19.21
4484670	20	DRY	23.066	1/31/16 14.35	2/1/16 21.55	S	13	2/1/16 22.18	2/1/16 0.00	YES			2/2/16 14.14	2/2/16 10.18	2/2/16 12.05	2/2/16 14.31
4484683	20	DRY	23.085	1/31/16 14.35	2/1/16 22.04	S	13	2/1/16 22.22	2/5/16 0.00					2/5/16 16.47	2/5/16 21.58	2/5/16 22.31
4484679	20	DRY	27.1	1/31/16 14.35	2/1/16 22.06	S	13	2/1/16 22.24	2/1/16 0.00					2/2/16 9.58	2/3/16 6.47	2/3/16 7.10
4484669	20	DRY	22.707	1/31/16 14.35	2/1/16 22.08	S	29	2/1/16 22.26	2/4/16 0.00	YES		2/4/16 15.43	2/5/16 16.11	2/4/16 10.53	2/6/16 5.46	2/6/16 6.06
4484663	20	DRY	13.608	1/31/16 14.35	2/1/16 22.11	S	29	2/1/16 22.27	2/4/16 0.00	YES				2/5/16 13.50	2/6/16 2.04	2/6/16 2.30
4484862	20	DRY	29.245	1/31/16 14.35	2/1/16 1.08	M	27	2/1/16 1.22	2/6/16 0.00					2/6/16 11.50	2/6/16 18.06	2/6/16 18.18
4485109	40	DRY	23.29	1/31/16 14.35	2/1/16 1.09	M	30	2/1/16 1.30	2/3/16 0.00					2/3/16 16.21	2/5/16 3.26	2/5/16 3.51
4484857	20	DRY	23.088	1/31/16 14.35	2/1/16 1.10	M	27	2/1/16 1.23	2/1/16 0.00	YES			2/2/16 14.13	2/2/16 10.18	2/2/16 17.56	2/2/16 19.31
4485103	40	DRY	11.028	1/31/16 14.35	2/1/16 1.10	M	30	2/1/16 1.31	2/3/16 0.00					2/3/16 17.08	2/9/16 14.05	2/9/16 14.35
4484856	20	DRY	29.855	1/31/16 14.35	2/1/16 1.11	M	27	2/1/16 1.24	2/3/16 0.00					2/3/16 16.44	2/5/16 5.56	2/5/16 6.07
4485102	40	DRY	30.139	1/31/16 14.35	2/1/16 1.13	M	30	2/1/16 1.27	2/15/16 0.00	YES				2/16/16 10.16	2/16/16 16.08	2/16/16 16.42
4485093	40	DRY	18.99	1/31/16 14.35	2/1/16 1.13	M	30	2/1/16 1.28	3/10/16 0.00	YES				3/10/16 9.43	3/10/16 13.12	3/10/16 13.52
4484849	20	DRY	29.67	1/31/16 14.35	2/1/16 1.14	M	51	2/1/16 1.58	2/2/16 0.00					2/4/16 10.12	2/5/16 14.03	2/5/16 14.43
4485092	40	DRY	29.166	1/31/16 14.35	2/1/16 1.15	M	30	2/1/16 1.32	2/15/16 0.00	YES				2/16/16 10.16	2/16/16 11.48	2/16/16 12.05
4485083	40	DRY	23.38	1/31/16 14.35	2/1/16 1.16	M	34	2/1/16 1.35	2/3/16 0.00					2/3/16 16.22	2/4/16 14.05	2/4/16 14.57
4485154	20	DRY	23.146	1/31/16 14.35	2/1/16 3.29	M	87	2/1/16 3.43	2/1/16 0.00	YES			2/2/16 14.13	2/2/16 10.18	2/2/16 17.19	2/2/16 17.42
4484747	40	DRY	11.978	1/31/16 14.35	2/1/16 3.29	N	32	2/1/16 3.39	2/3/16 0.00					2/3/16 16.23	2/5/16 2.10	2/5/16 2.39
4485157	20	DRY	23.408	1/31/16 14.35	2/1/16 3.30	M	87	2/1/16 3.43	2/24/16 0.00					2/26/16 9.43	2/26/16 10.02	2/26/16 10.14
4485149	20	DRY	24.912	1/31/16 14.35	2/1/16 3.33	M	89	2/1/16 3.45	2/2/16 0.00	YES			2/2/16 11.34	2/3/16 10.27	2/3/16 10.48	2/3/16 11.18
4484752	40	DRY	27.776	1/31/16 14.35	2/1/16 3.33	R	88	2/1/16 3.46	2/4/16 0.00					2/9/16 16.01	2/10/16 10.23	2/10/16 10.42
4484740	40	DRY	28.411	1/31/16 14.35	2/1/16 3.34	R	88	2/1/16 3.45	2/4/16 0.00					2/9/16 16.01	2/10/16 9.23	2/10/16 10.40
4485139	20	DRY	29.835	1/31/16 14.35	2/1/16 3.35	R	53	2/1/16 3.53	2/3/16 0.00					2/3/16 16.44	2/5/16 2.01	2/5/16 2.21
4485256	40	DRY	24.9	1/31/16 14.35	2/1/16 5.56	K	40	2/1/16 6.07	2/1/16 0.00					2/2/16 11.31	2/3/16 5.39	2/3/16 6.22
4484492	40	DRY	29.352	1/31/16 14.35	2/1/16 5.58	K	120	2/1/16 6.14	2/1/16 0.00	YES				2/2/16 12.10	2/4/16 5.02	2/4/16 6.59

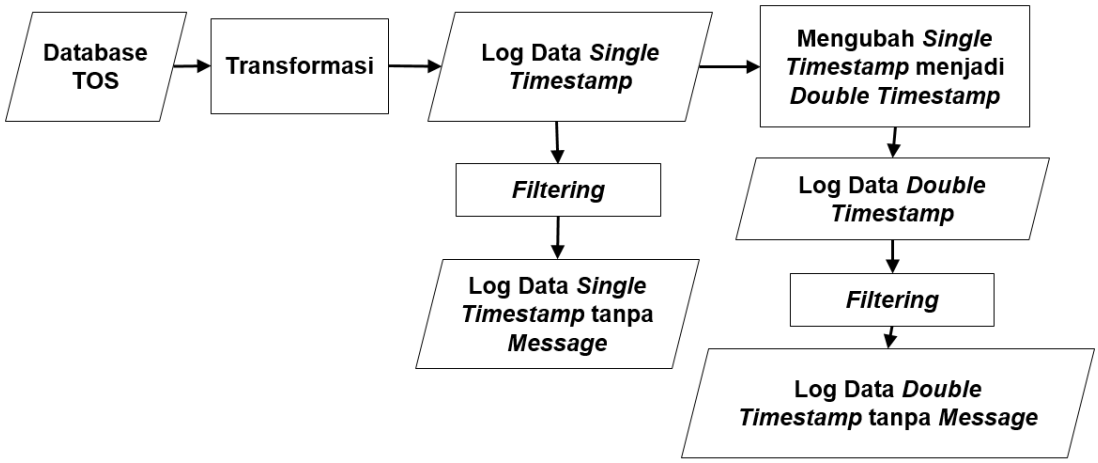
3.3 Pre-processing

3.3.1 Transformasi ke Bentuk Standar Data Log

Pada tahap ini adalah transformasi data PT TPS menjadi bentuk standar data log. Data log pada studi kasus ini memiliki komponen sebagai berikut:

- *Case* ID
- Aktivitas yang dijalankan pada proses bisnis
- *Message* atau pesan sebagai sarana komunikasi antar departemen
- *Sender* dan *receiver* dari setiap message yang terjadi
- *Originator* dari setiap aktivitas
- Waktu dijalankannya aktivitas dalam suatu *case* dalam proses bisnis (*timestamp*)
- *Input* dan *output* dokumen pada setiap aktivitas
- Biaya setiap aktitivas pada setiap *case* dalam proses bisnis
- Mesin yang digunakan pada setiap aktivitas
- *Yard Block*
- *Yard Slot*
- Detail lampiran

Dengan mengikuti arahan dari *expert* yakni Manajer IT PT PTS, diperoleh daftar aktivitas dan *message* lengkap dari awal hingga akhir untuk setiap *container* serta rentang waktu untuk setiap aktivitas. Total aktivitas dan *message* dari awal hingga akhir berjumlah 40. Tabel 3.3 akan dijelaskan semua aktivitas dan *message* dari data log PT TPS. Dan Gambar 3.3 menjelaskan alur pengolahan *database* menjadi data log.



Gambar 3.3 Alur pengolahan *database* menjadi data log

Tabel 3.3 Aktivitas dan *Message* dari data log PT TPS

No	Nama Aktivitas/*Message*
1	*Entry Document via PDE*

2	*RequestBehandle*
3	*RequestQuarantine*
4	*Vessel Berthing Process*
5	*Discharge Container*
6	*Bring Container to Yard*
7	*Stack Container in Yard*
8	*Approve Behandle*
9	*Verification Document Behandle*
10	*Create Job Order Document Behandle*
11	*SendJobOrderBehandleInfo*
12	*Bring Container from Yard to Behandle*
13	Stack Container in Behandle Area
14	Check Goods Behandle
15	Create document LHP
16	Stack Container in Yard From Behandle
17	*Create document SPPB*
18	*SendSPPBInfo*
19	*Approve Quarantine*
20	*Verification Document Quarantine*
21	*Create Job Order Document Quarantine*
22	*SendJobOrderQuarantineInfo*
23	*Bring Container from Yard to Quarantine*
24	*Stack Container in Quarantine Area*
25	*Check Goods Quarantine*
26	*Create document KH/KT*
27	*SendCertificateKH/KTInfo*
28	*Stack Container in Yard From Quarantine*
29	*Create Job Order Document Delivery*
30	*SendJobOrderDeliveryInfo*
31	*Truck in*
32	*Dispatch WQ Delivery to CHE*

33	*Determine Container Type*
34	*Determining Refeer/ Determining Dry/ Determining Uncointaner*
35	*Decide Task Before Lift Container*
36	*Unplug Refeer Cable/Prepare Tools*
37	*Lift on Container Truck*
38	*Truck Go To Gate Out*
39	*Check Container before Truck out*
40	*Truck Out*

Setelah memperoleh data nama aktivitas dan *message*, maka dengan menggunakan *rule* transformasi akan dihasilkan data log seperti pada Tabel 3.4. *Rule* transformasi yang digunakan untuk mengubah *database* menjadi data log standar dijelaskan pada Tabel 3.5. *Rule* transformasi bertujuan mengambil *database* sebagai *input file* dan mengolah *database* mengikuti *rule* yang telah ditetapkan sebelumnya yaitu dengan menggunakan intervensi dari *expert*, seperti lama waktu per aktivitas dan biaya per aktivitas dengan mengacu kepada SK Menteri Pemeriksaan Fisik Barang Impor (PDJBC Nomor 12 Tahun 2016), Keputusan Menteri Keuangan Nomor 453/KMK.04/2002 tentang Tatalaksana Kepabeanan di Bidang Impor dan Peraturan Menteri Perdagangan Nomor. 48/M-DAG/PER/7/2015 tentang Ketentuan Umum di Bidang Impor.

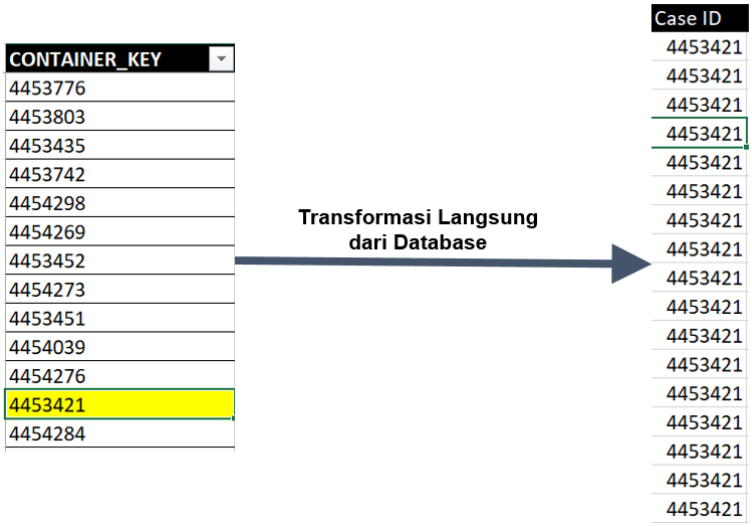
Tabel 3.4 Hasil Transformasi Data Log

Case ID	Sender	Originator	Input	Activity	Output	Receiver	Time	Cost	Mesin	Yard Block	Yard Slot	Detail Lampiran
4691694		Customer	NPWP, SIUP, API, SRP, TDP, NPIK, IT, INVOICE, PO, SK, BL, COO	Document_Entry_via_PDE	BC 2.0		24/03/2016 20.05	0	TOS			Dry;Green Line
4691694	Customer		BC 2.0	Request_Behandle	BC 2.0	SKP	24/03/2016 20.06	0				
4691694		TPS		Vessel_Berthing_Process			24/03/2016 22.10	48379,74				
4691694		TPS		Discharge_Container			24/03/2016 22.49	690,9082	CC			
4691694		TPS		Bring_Container_to_Yard			24/03/2016 22.57	74,99	HT Truck			
4691694		TPS		Stack_Container_in_Yard			24/03/2016 22.59	28,99	RTGC	R	106	
4691694	SKP		BC 2.0	Approve_Behandle	BC 2.0	Customer	26/03/2016 04.16	1,358024				
4691694		SKP	BC 2.0	Verification_Document_Behandle	BC 2.0		27/03/2016 08.39	8,604293				Dry;Green Line
4691694		Pejabat Bea Cukai	LHP, BC 2.0, Berita Acara	Create_document_SPPB	SPPB		27/03/2016 22.48	2055,586				
4691694	Pejabat Bea Cukai		SPPB	Send_SPPB_Info		Customer	27/03/2016 22.50	0				

Tabel 3.5 Potongan *rule* transformasi

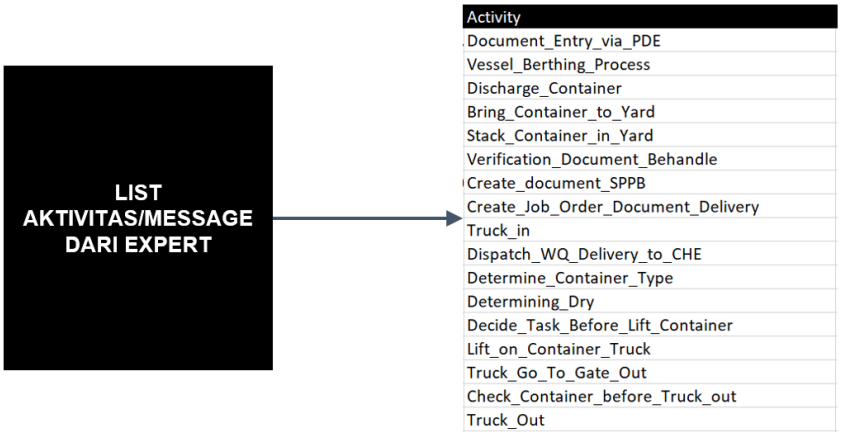
```
INSERT INTO `table 4`
(Case_id,Sender,Originator,Input,Activity,Output,Receiver,actTime,Cost,Yard_block,Yard_s
lot,Detail_lampiran)
SELECT CONTAINER_KEY as Case_id,'' as Sender,'Customer' as Originator, 'NPWP,
SIUP, API, SRP, TDP, NPIK, IT, INVOICE, PO, SK, BL, COO' as
Input,'Document_Entry_via_PDE' as Activity,  'BC 2.0' as Output,'' as Receiver,(
VESSEL_ATB-((RAND()*(0.125-0.08333333))+0.08333333) ) AS Time,'0' as Cost,'' as
Yard_block,'' as Yard_slot,
(IF(HAS_QUARANTINE_FLAG ='YES',
(case when CTR_TYPE='DRY' && CUSTOMS_BEHANDLE_COUNT=0 then
'Quarantine;Dry;Green Line'
        when CTR_TYPE='DRY' && CUSTOMS_BEHANDLE_COUNT=1 then
'Quarantine;Dry;Red Line'
        when CTR_TYPE='RFR' && CUSTOMS_BEHANDLE_COUNT=0 then
'Quarantine;Reefer;Green Line'
        when CTR_TYPE='RFR' && CUSTOMS_BEHANDLE_COUNT=1 then
'Quarantine;Reefer;Red Line'
        when CTR_TYPE='OVD' && CUSTOMS_BEHANDLE_COUNT=0 then
'Quarantine;Uncontainer;Green        Line'
        when CTR_TYPE='OVD' && CUSTOMS_BEHANDLE_COUNT=1 then
'Quarantine;Uncontainer;Red  Line'
        ELSE ''
        END
) ,(case when CTR_TYPE='DRY' && CUSTOMS_BEHANDLE_COUNT=0 then
'Dry;Green Line'
        when CTR_TYPE='DRY' && CUSTOMS_BEHANDLE_COUNT=1 then
'Dry;Red Line'
        when CTR_TYPE='RFR' && CUSTOMS_BEHANDLE_COUNT=0 then
'Reefer;Green Line'
        when CTR_TYPE='RFR' && CUSTOMS_BEHANDLE_COUNT=1 then
'Reefer;Red Line'
        when CTR_TYPE='OVD' && CUSTOMS_BEHANDLE_COUNT=0 then
'Uncontainer;Green Line'
        when CTR_TYPE='OVD' && CUSTOMS_BEHANDLE_COUNT=1 then
'Uncontainer;Red Line'
        ELSE ''
        END
) )) as Detail_lampiran
FROM `table 1` ;

INSERT INTO `table 4`
(Case_id,Sender,Originator,Input,Activity,Output,Receiver,actTime,Cost,Yard_block,Yard_s
lot,Detail_lampiran)
SELECT Case_id as Case_id,'Customer' as Sender,'' as Originator, 'BC 2.0' as
Input,'Request_Behandle' as Activity,  'BC 2.0' as Output,'SKP' as Receiver,(
actTime+((RAND()*(0.001388889-0.000694444))+0.000694444) )
 AS Time,'0' as Cost,'' as Yard_block,'' as Yard_slot,'' as Detail_lampiran
FROM `table 4`
where Activity='Document_Entry_via_PDE';
```

```
INSERT INTO `table 4`
(Case_id,Sender,Originator,Input,Activity,Output,Receiver,actTime,Cost,Yard_block,Yard_slot,Detail_lampiran)
SELECT Case_id as Case_id,'Customer' as Sender,'' as Originator, 'COO, Health Certificate, Sanitary Certificate, CITES, SPI, BL, Invoice, Packing List, Cargo Manifest, DO' as Input, 'Request_Quarantine' as Activity,'KH-1/KT-1' as Output,'Petugas Karantina' as Receiver, ( actTime+((RAND()*(0.001388889-0.000694444))+0.000694444) ) as Time,'0' as Cost,'' as Yard_block,'' as Yard_slot,'' as Detail_lampiran
FROM `table 4`
where Activity='Request_Behandle';
.
.
.
ORDER BY Case_id,Time
```

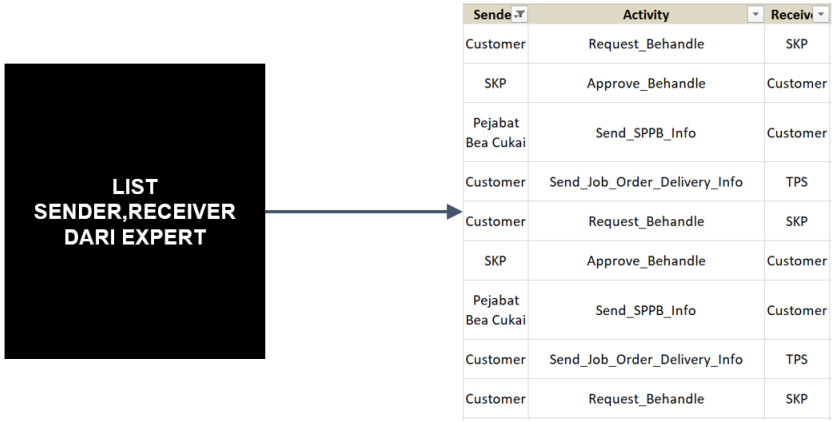


Gambar 3.4 *Case ID* pada data log

Gambar 3.4 menjelaskan transformasi dari *Case ID*. *Case ID* data log diperoleh dari *container key* pada *database* PT TPS. Gambar 3.5 menjelaskan transformasi dari aktivitas dan *message*. *List* aktivitas dan *message* diperoleh dari arahan *expert*.



Gambar 3.5 Aktivitas dan *Message* pada data log



Sende	Activity	Receiv
Customer	Request_Behandle	SKP
SKP	Approve_Behandle	Customer
Pejabat Bea Cukai	Send_SPPB_Info	Customer
Customer	Send_Job_Order_Delivery_Info	TPS
Customer	Request_Behandle	SKP
SKP	Approve_Behandle	Customer
Pejabat Bea Cukai	Send_SPPB_Info	Customer
Customer	Send_Job_Order_Delivery_Info	TPS
Customer	Request_Behandle	SKP

Gambar 3.6 *Sender* dan *Receiver* pada data log

Gambar 3.6 menjelaskan transformasi dari *sender* dan *receiver* pada data log. *Sender* dan *receiver* juga diperoleh dari arahan *expert.* Gambar 3.7 menjelaskan transformasi dari *originator*. Sama seperti *sender* dan *receiver*, *originator* juga diperoleh dari informasi *expert.*

Originator	Activity
Customer	Document_Entry_via_PDE
TPS	Vessel_Berthing_Process
TPS	Discharge_Container
TPS	Bring_Container_to_Yard
TPS	Stack_Container_in_Yard
SKP	Verification_Document_Behandle
Pejabat Bea Cukai	Create_document_SPPB
Customer	Create_Job_Order_Document_Delivery
Customer	Truck_in
TPS	Dispatch_WQ_Delivery_to_CHE
TPS	Determine_Container_Type
TPS	Determining_Dry
TPS	Decide_Task_Before_Lift_Container
TPS	Lift_on_Container_Truck
Customer	Truck_Go_To_Gate_Out
TPS	Check_Container_before_Truck_out
Customer	Truck_Out
Customer	Document_Entry_via_PDE

Gambar 3.7 *Originator* pada data log

CONTAINER_	CTR_TYPE	HAS_QUARAN	TRUCK_IN_DA	TRUCK_OUT_D	CUSTOMS_BEHA
4454039	DRY		1/26/16 18.12	1/26/16 20.39	0
4454276	DRY		1/27/16 22.41	1/27/16 23.05	0
4453421	DRY		1/27/16 15.34	1/27/16 16.33	0

inversi distribusi kumulatif normal

Case ID	Activity	Time
4453421	Truck_in	27/01/2016 15.34
4453421	Dispatch_WQ_Delivery_to_CHE	27/01/2016 15.35
4453421	Determine_Container_Type	27/01/2016 15.38
4453421	Determining_Dry	27/01/2016 15.40
4453421	Decide_Task_Before_Lift_Container	27/01/2016 15.42
4453421	Lift_on_Container_Truck	27/01/2016 15.44
4453421	Truck_Go_To_Gate_Out	27/01/2016 16.30
4453421	Check_Container_before_Truck_out	27/01/2016 16.31
4453421	Truck_Out	27/01/2016 16.33

Gambar 3.8 *Timestamp* pada data log

Gambar 3.8 menunjukkan transformasi untuk *timestamp* PT TPS. Perhitungan *timestamp* dilakukan dengan rumus matematika berdasarkan arahan *expert* karena dari *database* PT TPS yang waktunya hanya direkam per kelompok aktivitas akan dibagi menjadi bebeapa aktivitas di data log. Berikut rumus yang digunakan untuk transformasi *timestamp:*

$$\text{time}(Act\_now) = time(Act\_before) + \text{NormInv}(\text{p}, \mu, \sigma_{\text{x}})$$

$$\mu = \frac{\text{median}(\text{Es. Act})}{\sum_{i=0}^{n} \text{median}(\text{Es. Act}_{\text{i}})} \times \text{ActInLog}$$

$$\sigma_x = 0{,}05 \times \mu$$

Keterangan:

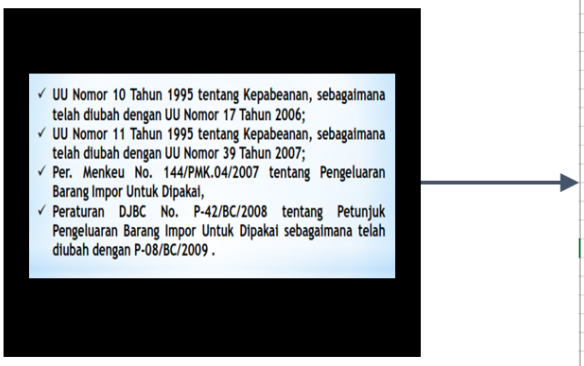
time(Act_now) = waktu aktivitas sekarang

time(Act_before) = waktu aktivitas sebelumnya

Es.Act = estimasi *sojourn time* expert

ActInLog = rentang waktu kelompok aktivitas pada database TOS

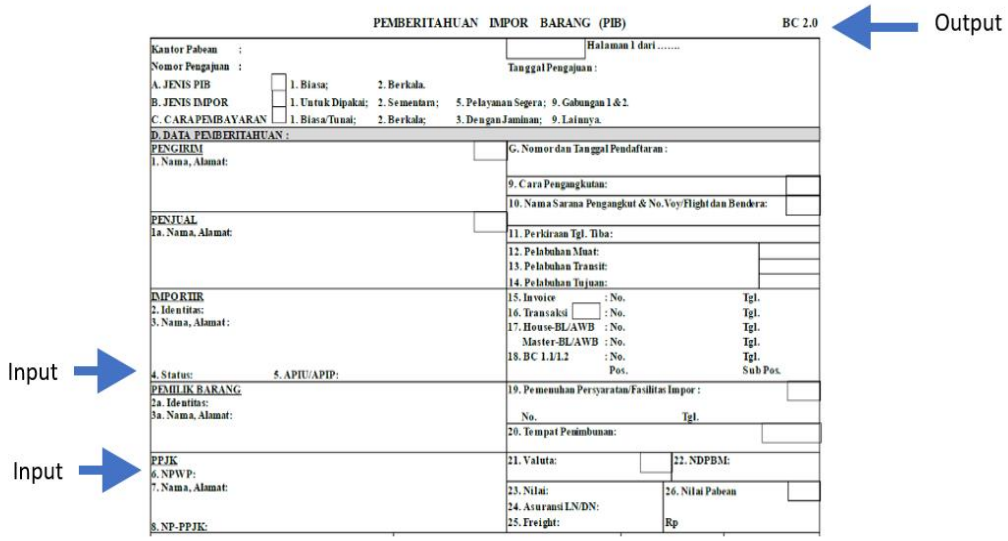
n = jumlah aktivitas pada kelompok aktivitas



Case ID	Input	Activity	Output
4453421	NPWP, SI	Document_Entry_via_PDE	BC 2.0
4453421	BC 2.0	Request_Behandle	BC 2.0
4453421		Vessel_Berthing_Process	
4453421		Discharge_Container	
4453421		Bring_Container_to_Yard	
4453421		Stack_Container_in_Yard	
4453421	BC 2.0	Approve_Behandle	BC 2.0
4453421	BC 2.0	Verification_Document_Behandle	BC 2.0
4453421	LHP, BC 2.	Create_document_SPPB	SPPB
4453421	SPPB	Send_SPPB_Info	
4453421	SPPB	Create_Job_Order_Document_Delivery	CEIR
4453421		Send_Job_Order_Delivery_Info	
4453421		Truck_in	
4453421		Dispatch_WQ_Delivery_to_CHE	
4453421		Determine_Container_Type	
4453421		Determining_Dry	
4453421		Decide_Task_Before_Lift_Container	
4453421		Lift_on_Container_Truck	
4453421		Truck_Go_To_Gate_Out	
4453421		Check_Container_before_Truck_out	
4453421		Truck_Out	

Gambar 3.9 *Input* dan *output* pada data log

Gambar 3.9 menunjukkan transformasi untuk *input* dan *output* dari *database* PT TPS ke dalam bentuk data log. *Input* dan *output* ini berdasarkan dari undang-undang dan peraturan DJBC seperti pada Gambar 3.10.



Gambar 3.10 Penentuan *Input* dan *output* pada data log

CONTAINER_	CTR_TYPE	HAS_QUARAN	CUSTOMS_BEH/
4454039	DRY		0
4454276	DRY		0
4453421	DRY		0

Case ID	Activity	Detail Attachment
4453421	Document_Entry_via_PDE	Dry
4453421	Vessel_Berthing_Process	
4453421	Discharge_Container	
4453421	Bring_Container_to_Yard	
4453421	Stack_Container_in_Yard	
4453421	Verification_Document_Behandle	Green Line
4453421	Create_document_SPPB	
4453421	Create_Job_Order_Document_Delivery	
4453421	Truck_in	

Gambar 3.11 Detail *attachment* pada data log

Penentuan detail *attachment* untuk transformasi *database* menjadi data log pada Gambar 3.11 didapatkan dari data pada *database* yang dapat mendukung informasi aktivitas seperti pada Gambar 3.12.

Detail Attachment	Pengambilan dari Database
Dry	If CTR_TYPE == DRY
Reefer	If CTR_TYPE == RFR
Uncontainer	If CTR_TYPE != DRY and CTR_TYPE != RFR
Green Line	If CUSTOM_BEHANDLE_COUNT == 0
Red Line	If CUSTOM_BEHANDLE_COUNT == 1
Quarantine	If HAS_QUARANTINE == YES

Gambar 3.12 Penentuan detail *attachment* pada data log

Gambar 3.13 *Cost* pada data log

NO	URAIAN	MASA	BARANG UMUM TON/M3 (Rp.)	BRG. MENGGANGGU TON/M3 (Rp.)	BARANG KIMIA TON/M3 (Rp.)	BRG. BERBAHAYA TON/M3 (Rp.)	GT / ETMAL
1	PENUMPUKAN BARANG	I (Hari 1 -10)	1,250	1,875	1,875	2,500	
		II (Hari 11 dst)	2,500	3,750	3,750	5,000	
2	UANG DERMAGA		2,550	3,825	3,825	5,100	
3	DELIVERY		9,212	11,055	13,818	18,424	

Gambar 3.14 Potongan *Cost* pada data tarif pelayanan bongkar muat peti kemas TPS

Pada Gambar 3.13 dijelaskan transformasi *cost* yang digunakan pada data log ini. *Cost* diperoleh dari data taris pelayanan bongkar buat peti kemas TPS seperti pada Gambar 3.14. Untuk *cost* sama seperti halnya *time*, yang tersedia hanya *cost* per kelompok aktivitas, maka digunakan rumus untuk membagi *cost* per kelompok aktivitas menjadi *cost* per aktivitas. Rumus yang digunakan sebagai berikut:

$$cost(Act) = \frac{time(Act)}{time(GroupAct)} \times cost(GroupAct)$$

Keterangan:

cost(Act) = biaya Aktivitas

cost(GroupAct) = biaya kelompok aktivitas

time(Act) = *sojourn time* aktivitas

time(GroupAct) = rata-rata *sojourn time* kelompok aktivitas

3.3.2 *Filtering* antara Aktivitas dan *Message*

Dari Tabel 3.6, selanjutnya akan difilter antara aktivitas dan *message* dimana hal ini bertujuan untuk *proses discovery*, scheduling dan optimasi. *Proses discovery* dengan algoritma modifikasi *Alpha* memodelkan hanya aktivitas saja, begitu pula untuk *scheduling* dan optimasi. Proses *scheduling* dan optimasi hanya melibatkan aktivitas sehingga dilakukan proses *filtering* antara aktivitas dan *message*. Proses *filterin*g dilakukan dengan membuat *rule* jika terdapat *sender* dan *receiver* maka dikategorikan sebagai *message*, dan sebaliknya jika terdapat *originator* maka dikategorikan sebagai aktivitas. Seluruh aktivitas pada data log PT TPS berjumlah 31.

```
1.  activity_or_message = data_in_column_activity_of_Log
2.  for x in activity_or_message:
3.      if sender(x) and receiver(x) != NULL:
4.          Log_message.add(message)
5.          Log_data.erase(message)
```

Tabel 3.6 Seluruh aktivitas pada data log PT TPS

No	Aktivitas
1	*Entry Document via PDE*
2	*Vessel Berthing Process*
3	*Discharge Container*
4	*Bring Container to Yard*
5	*Stack Container in Yard*
6	*Verification Document Behandle*
7	*Create Job Order Document Behandle*
8	*Bring Container from Yard to Behandle*
9	*Stack Container in Behandle Area*
10	*Check Goods Behandle*
11	*Create document LHP*
12	*Stack Container in Yard From Behandle*
13	*Create document SPPB*
14	*Verification Document Quarantine*
15	*Create Job Order Document Quarantine*
16	*Bring Container from Yard to Quarantine*
17	*Stack Container in Quarantine Area*
18	*Check Goods Quarantine*
19	*Create document KH/KT*
20	*Stack Container in Yard From Quarantine*
21	*Create Job Order Document Delivery*
22	*Truck in*
23	*Dispatch WQ Delivery to CHE*
24	*Determine Container Type*
25	*Determining Refeer/ Determining Dry/ Determining Uncointaner*

26	*Decide Task Before Lift Container*
27	*Unplug Refeer Cable/Prepare Tools*
28	*Lift on Container Truck*
29	*Truck Go To Gate Out*
30	*Check Container before Truck out*
31	*Truck Out*

3.3.3 Pengelompokan Aktivitas Berdasarkan Kategori Dokumen dan Barang

Setelah diperoleh keseluruhan aktivitas di PT TPS maka selanjutnya adalah menentukan aktivitas tersebut adalah kategori dokumen atau barang. Maksud kategori barang atau dokumen adalah proses yang dilakukan oleh suatu aktivitas adalah terkait dokumen atau barang (*container*). Dapat diketahui dengan melihat *input* dan *output* pada data log. Dengan menggunakan *rule* akan dihasilkan aktivitas dan kategorinya seperti pada Tabel 3.7.

Tabel 3.7 Pemisahan aktivitas berdasarkan kategori barang dan dokumen

No	Aktivitas	Kategori
1	*Entry Document via PDE*	Dokumen
2	*Vessel Berthing Process*	Barang
3	*Discharge Container*	Barang
4	*Bring Container to Yard*	Barang
5	*Stack Container in Yard*	Barang
6	*Verification Document Behandle*	Dokumen
7	*Create Job Order Document Behandle*	Dokumen
8	*Bring Container from Yard to Behandle*	Barang
9	*Stack Container in Behandle Area*	Barang
10	*Check Goods Behandle*	Barang
11	*Create document LHP*	Dokumen
12	*Stack Container in Yard From Behandle*	Barang
13	*Create document SPPB*	Dokumen
14	*Verification Document Quarantine*	Dokumen
15	*Create Job Order Document Quarantine*	Dokumen
16	*Bring Container from Yard to Quarantine*	Barang
17	*Stack Container in Quarantine Area*	Barang
18	*Check Goods Quarantine*	Barang
19	*Create document KH/KT*	Dokumen
20	*Stack Container in Yard From Quarantine*	Barang
21	*Create Job Order Document Delivery*	Dokumen
22	*Truck in*	Barang
23	*Dispatch WQ Delivery to CHE*	Barang
24	*Determine Container Type*	Barang
25	*Determining Refeer/ Determining Dry/ Determining Uncointaner*	Barang
26	*Decide Task Before Lift Container*	Barang
27	*Unplug Refeer Cable/Prepare Tools*	Barang
28	*Lift on Container Truck*	Barang
29	*Truck Go To Gate Out*	Barang

30	*Check Container before Truck out*	Barang
31	*Truck Out*	Barang

3.3.4 Membuat *Single Timestamp* menjadi *Double Timestamp*

Data log dapat dibagi menjadi dua jenis berdasarkan waktu yang terekam. Jenisnya adalah *single timestamp* dan *double timestamp*. Dalam organisasi yang menjalankan proses bisnis, data log tersedia baik dalam bentuk *single timestamp* atau *double timestamp*. Jika organisasi menyediakan waktu mulai dan waktu selesai untuk semua kegiatan yang dijalankan, data log ini dapat langsung digunakan untuk menemukan model proses dengan algoritma usulan yaitu modifikasi *Time-based Alpha Miner*. Namun ada organisasi yang hanya menyediakan waktu akhir untuk semua aktivitas yang dieksekusi, maka hal pertama yang perlu dilakukan adalah mengubah *single timestamp* menjadi *double timestamp* dengan menggunakan *sojourn time*. *Sojourn time* adalah adalah total waktu (termasuk durasi eksekusi dan waktu tunggu) yang dibutuhkan setiap aktivitas untuk menyelesaikan proses. Langkah-langkah untuk mengubah *single timestamp* dan *double timestamp* adalah sebagai berikut:

1. Hitung *sojourn time* untuk semua aktivitas di data log
$$Sojourn\ time\ B = et_{activity_B} - et_{activity_A}$$
2. Pilih batas atas dan batas bawah setiap aktivitas dari *sojourn time*
3. Hitung nilai median antara batas atas dan batas bawah untuk setiap aktivitas
4. Lakukan normalisasi untuk setiap aktivitas
$$Normalization = \frac{Median\ value}{Average\ of\ Sojourn\ Time}$$
5. Hitung standar deviasi untuk setiap aktivitas
$$Stdev = \frac{0.05}{Normalization}$$
6. Dapatkan durasi eksekusi untuk setiap aktivitas dengan menggunakan bilangan acak normal
$$Norminv = Rand(); normalization; stdev$$
7. Untuk mendapatkan waktu mulai untuk setiap aktivitas, kurangkan waktu akhir dengan durasi eksekusi dan untuk mendapatkan waktu tunggu, kurangkan waktu akhir dengan waktu mulai aktivitas.
$$Start\ time\ act\ B = End\ time\ B - Execution\ duration\ B$$
$$Waiting\ time\ act\ B = End\ time\ B - Start\ time\ C$$

Misalnya, terdapat data log dengan *single timestamp* seperti pada Tabel 3.8. Dengan menggunakan langkah 1-7, kita dapat mengubah *single timestamp* menjadi *double timestamp* dengan menggunakan *sojourn time*. Kami menerapkan langkah-langkah untuk mendapatkan durasi eksekusi untuk aktivitas B dan aktivitas C serta mengurangi waktu akhir dengan durasi eksekusi untuk mendapatkan waktu mulai untuk aktivitas B dan aktivitas C. Tabel 3.9 menyajikan data log *double timestamp* sebagai hasilnya.

Tabel 3.8 Langkah-langkah mengubah *single timestamp* menjadi *double timestamp*

Case ID	Activity	End Time	Sojourn Time	Average Sojourn Time	Upper bound	Lower bound	Median	Normalization	Stdev	Execution Duration
ID001	A	20/06/2014 08.32	03.17.00							
ID001	B	20/06/2014 13.42	05.10.18	06.36	09.59	05.05	07.32	03.22	01.22	04.39
ID001	B	20/06/2014 23.41	09.59.22							04.18
ID001	C	21/06/2014 08.16	08.34.09	10.16	15.35	06.41	11.08	01.59	01.18	01.54
ID002	A	21/06/2014 10.46	02.30.18							
ID002	B	21/06/2014 16.57	06.11.36							03.09
ID002	D	21/06/2014 18.09	01.11.27							
ID002	E	21/06/2014 19.15	01.06.29							
ID002	C	22/06/2014 10.51	15.35.29							01.16
ID003	A	22/06/2014 16.28	05.36.48							
ID003	D	22/06/2014 23.42	07.14.33							
ID003	B	23/06/2014 04.48	05.05.50							03.10
ID003	E	23/06/2014 16.44	11.56.24							
ID003	C	23/06/2014 23.26	06.41.04							03.54

Tabel 3.9 Hasil data log dengan *double timestamp*

Case ID	Activity	Start Time	End Time
ID001	A		20/06/2014 08.32
ID001	B	20/06/2014 09.03	20/06/2014 13.42
ID001	B	20/06/2014 19.23	20/06/2014 23.41
ID001	C	21/06/2014 06.22	21/06/2014 08.16
ID002	A		21/06/2014 10.46
ID002	B	21/06/2014 13.48	21/06/2014 16.57
ID002	D		21/06/2014 18.09
ID002	E		21/06/2014 19.15
ID002	C	22/06/2014 09.34	22/06/2014 10.51
ID003	A		22/06/2014 16.28
ID003	D		22/06/2014 23.42
ID003	B	23/06/2014 01.38	23/06/2014 04.48
ID003	E		23/06/2014 16.44
ID003	C	23/06/2014 19.31	23/06/2014 23.26

3.4 *Proses Discovery* dengan *Modified Time-based Alpha Miner*

3.4.1 Temporal Causal Relation

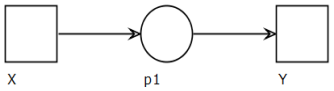
Enam jenis *temporal causal relation* antar aktivitas didefinisikan sebagai versi perpanjangan dari relasi yang terdapat pada data log. Penelitian ini menyajikan konsep *double timestamp* untuk menemukan hubungan model proses bisnis berdasarkan interval berbasis waktu. Berdasarkan *temporal causal relation*, untuk relasi sekuensial, dikelompokkan menjadi *before* dan *meets*. Sementara itu, untuk hubungan paralel akan

dikelompokkan menjadi *overlaps*, *contains*, *equals*, *has the same end time*, dan *has the same start time*.

Definisi 3.1 Relasi *causal* menunjukkan relasi paralel antara dua aktivitas *X* (X_s, X_f) dan *Y* (Y_s, Y_f) dimana X_s sebagai waktu mulai eksekusi aktivitas X dan X_f sebagai waktu selesai eksekusi aktivitas X, serta terdapat data log (*L*) dan *trace* (σ) sehingga $\sigma \in L$ dan $X, Y \in L$ dapat dibedakan sebagai berikut:

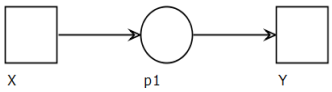
- $Before,\ X > Y\ iff\ X_f \leq Y_s$

X	6/20/14 8.32	6/20/14 11.29	X before Y
Y	6/20/14 13.42	6/20/14 20.12	



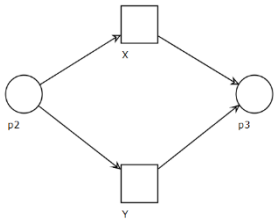
- $Meets,\ X > Y\ iff\ X_f \leq Y_s$

X	6/20/14 23.41	6/21/14 6.41	X meets Y
Y	6/21/14 6.41	6/21/14 9.55	



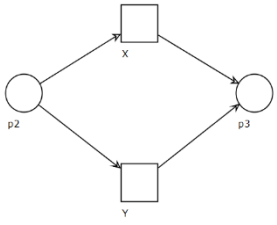
- $Overlaps,\ X \square Y\ \ iff\ X_f > Y_s\ \wedge\ X_f < Y_f$

X	7/16/14 23.21	7/17/14 9.54	X overlaps Y
Y	7/17/14 8.43	7/17/14 21.35	



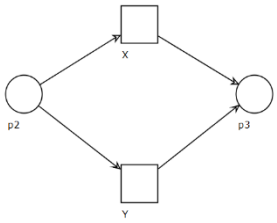
- $Contains,\ X@Y\ iff\ X_s < Y_s\ \wedge\ Y_f > X_s\ \wedge\ \ X_f > Y_f$

X	7/2/14 4.30	7/3/14 3.06	X contains Y
Y	7/2/14 14.50	7/3/14 1.44	



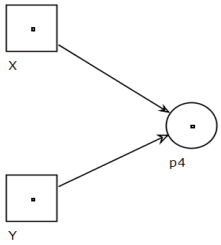
- $Equals,\ X \Diamond Y\ iff\ X_s = Y_s\ \wedge\ X_f = Y_f$

X	7/24/14 6.59	7/25/14 0.20	X equals Y
Y	7/24/14 6.59	7/25/14 0.20	



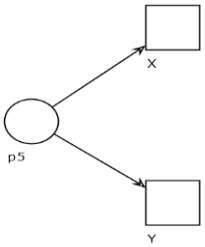
- $Has\ the\ same\ end\ time,\ X_f Y\ iff\ X_f = Y_f \wedge\ X_s < Y_s\ \wedge\ Y_s < X_f$

X	6/22/14 10.51	6/22/14 23.42	X has the same end time Y
Y	6/22/14 16.28	6/22/14 23.42	



- $Has\ the\ same\ start\ time,\ X_p Y\ iff\ X_s = Y_s\ \wedge\ X_f > Y_f$

X	7/9/14 14.38	7/10/14 13.01	X has the same start time Y
Y	7/9/14 14.38	7/10/14 9.19	



3.4.2 *Control Flow* dan *Double Timestamp* dalam Instans Proses

Untuk membuat model proses yang ditemukan dapat dengan jelas menunjukkan *control flow* antara aktivitas dan hubungan temporal mereka (yaitu, hubungan paralel dan *sequence*), maka definisi formal tentang *control flow* adalah suatu keharusan.

Definis 3.2 *Control flow* antara dua aktivitas $X(X_s,X_f)$ dan $Y(Y_s,Y_f)$ dimana terdapat data log (L) dan *trace* (σ) sehingga $\sigma \in L$ dan $X,Y \in L$ dapat dibedakan sebagai berikut:

- $Sequence,\ X \rightarrow Y\ \ iff\ X > Y \wedge Y \square X$

- $Not\ related$, X#Y $iff\ X \square Y \wedge Y \square X$
- $Parallel,\ X \parallel Y\ \ iff\ X > Y\ \wedge Y > X\ \vee \{X \square Y\ \vee\ X@Y\ \vee X_f Y\ \vee X \Diamond Y\ \vee X_p Y\}$
- $Conditional\ XOR,\ X \otimes Y\ if\ there\ exist\ only\ X\ or\ Y\ in\ any\ traces\ in\ the\ \log(L)$
- $Parallel\ AND,\ X \bullet Y\ \ iff\ X \parallel$
  $Y\ and\ there\ not\ exist\ X \otimes Y\ in\ any\ traces\ in\ the\ \log(L)$
- $Conditional\ OR,\ X \oplus Y\ \ iff\ X \parallel Y\ and\ there\ exist\ X \otimes Y\ in\ any\ traces\ in\ the\ \log(L)$

Definisi 3.1 *Double timestamp* dalam Instans Proses.
Diberikan *data log L* dan *trace* σ sehingga σ∈*L*. Relasi *sequence* A→B dan A→C antaraktivitas $A(e_s,e_f)$, $B(e_s,e_f)$, dan $C(e_s,e_f)$ menyebabkan A>B, A>C, dan B||C. Begitu juga relasi *sequence* B→D dan C→D antaraktivitas $A(e_s,e_f)$, $B(e_s,e_f)$, dan $C(e_s,e_f)$ menyebabkan B>D, C>D, dan B||C.

3.4.3 Algoritma *Modified Time-based Alpha Miner*

Bagian ini menjelaskan langkah-langkah algoritma modifikasi dalam menemukan model proses menggunakan Definisi 3.1, Definisi 3.2 dan Definisi 3.3. Langkah utamanya adalah, pertama, mengelompokkan semua *input* dan *output* aktivitas untuk setiap *trace*; kemudian, klasifikasi relasi *sequence* dan paralel, dan terakhir, model proses lengkap dengan relasi *sequence* dan paralel, *input* dan *output.*

Step 1. Kelompokkan relasi *sequence* (>) setiap aktivitas di seluruh *case* di data log

Step 2. Kelompokkan relasi paralel (||) setiap aktivitas di seluruh *case* di data log

Step 3. Hapus relasi *sequence* dan paralel yang duplikat untuk mendapatkan jumlah *trace* di data log

Step 4. Mendapatkan semua *trace* dengan relasi paralel dan *sequence* dari data log

Step 5. Menentukan transisi (T_L) dari *workflow net*

$$\boldsymbol{T_L = \{t\ \in T \mid \exists_{\sigma \epsilon L} t\ \in \sigma\}}$$

Step 6. Menentukan transisi *input* dari *workflow net*

$$\boldsymbol{T_I = \{t\ \in T \mid \exists_{\sigma \epsilon L} t = first(\sigma)\}}$$

Step 7. Menentukan transisi *output* dari *workflow net*

$$\boldsymbol{T_o = \{t\ \in T \mid \exists_{\sigma \epsilon L} t = last(\sigma)\}}$$

Step 8. Membuat *gantt chart* untuk semua *trace* berdasarkan *temporal causal relation* dan *control flow* untuk menunjukkan hubungan antara semua aktivitas dalam *trace* secara jelas

Step 9. Membuat *place*

$$\boldsymbol{P_L = \{p_{(A,B)}\ \mid (A,B) \in Y_L\} \cup \{i_L, o_L\}}$$

Step10. Menemukan model proses berdasarkan relasi *sequence* dan paralel

Step11. Klasifikasikan kemungkinan tipe relasi paralel (AND dan OR)

$$X \bullet Y,\ X \oplus Y\ iff\ X \parallel Y\ and\ there\ exist\ Z \in L\ where\ X > Z\ \vee\ Y > Z\ \vee\ X \square Z\ \vee Y \square Z\ in\ any\ \sigma \in L$$

Parallel AND

$$foreach\ L\ in \bullet,\ which\ (X,Y) \wedge (Z,A) \in L$$

$$iff\ X = [Y, (Z \wedge A)] \wedge [(Y,\ (A \wedge Z)] \wedge$$

$$[(Z \wedge A),\ Y]\ \wedge\ [(A \wedge Z),\ Y)] \in\ >_L\ then\ [A, (B, (C \wedge D))]$$

Conditional OR

$$foreach\ L\ in \oplus,\ which\ (X,Y) \wedge (Z,A) \in L$$

$$iff\ X = (Z \vee A) \wedge [(Z \vee A), Y] \vee$$

$$[Y, (Z \vee A)] \in\ >_L\ then\ [X, (Y, (Z \vee A))]$$

Step 12. Tambahkan relasi *sequence* dan *input-output* kedalam model proses

$$foreach\ R\ in\ >_L,\ which\ (X,Y) \in L$$

$$iff\ (X,Y) \notin G$$

$$iff\ (\bullet Y) not\ exist$$

$$G \longleftarrow G \cup (X,Y)$$

$$else\ iff\ X \bullet Z,\ which\ (Z,Y) \in G$$

$$G \longleftarrow G \cup [(X,Z) \bullet B]$$

$$else\ iff\ X \oplus Z,\ which\ (Z,Y) \in G$$

$$G \longleftarrow G \cup [(X,Z) \oplus B]$$

$$else\ G \longleftarrow G \cup [(X,Z) \otimes B]$$

Step 13. Model proses dalam bentuk Petri Net selesai

$$\boldsymbol{\alpha(L) = (P_L,\ T_L, F_L)}$$

3.4.4 Penentuan Relasi AND, OR dan XOR

Perhitungan relasi paralel membutuhkan frekuensi aktivitas paralel di data log. Frekuensi diperoleh dengan menghitung hubungan paralel dan *sequence* masing-masing aktivitas di data log.

Interval untuk XOR, OR, dan AND adalah sebagai berikut:

- XOR
  $If\ Avg\ PM\ \leq Minimum\ All\ Sequence\ Relation, then\ XOR$
- OR
  $If\ Minimum\ All\ Sequence\ Relation \leq Avg\ PM\ \leq Avg\ All\ Sequence\ Relation, then\ OR$
- AND
  $If\ Avg\ All\ Sequence\ Relation\ \leq Avg\ PM, then\ AND$

Keterangan:

$Avg\ PM$: rata-rata dari *parallel measure* yang memiliki induk aktivitas yang sama, baik *undirect and direct followed frequency* aktivitas

3.5 Paralelisasi Aktivitas

Untuk memparalelkan sebuah proses bisnis, aktivitas yang independen harus diidentifikasi terlebih dahulu, seperti aktivitas manakah dari proses bisnis yang dapat dilakukan bersamaan. Tingkat paralelisme tertinggi dicapai jika jumlah aktivitas yang diidentifikasi sebagai independen dapat dimaksimalkan. Secara umum, identifikasi ini didasarkan pada waktu dan tempat eksekusi aktivitas, aktivitas dapat di paralelisasi jika aktivitas pada entitas simulasi yang sama dijalankan dalam urutan timestamp. Untuk meningkatkan tingkat paralelisme, kami mengusulkan sebuah pendekatan baru yang mengidentifikasi kriteria independensi lain: Jika dua aktivitas pada entitas simulasi yang sama mengakses item data yang sama dengan cara yang berbeda, mereka dapat dieksekusi secara paralel.

Kriteria aktivitas dapat diparalelkan adalah sebagai berikut:

- Aktivitas yang independen harus diidentifikasi terlebih dahulu
- Aktivitas dapat di paralelisasi jika aktivitas pada entitas simulasi yang sama dijalankan dalam urutan timestamp
- Jika dua aktivitas pada entitas simulasi yang sama mengakses item data yang sama dengan cara yang berbeda
- Aktivitas setiap departemen independen
- Waktu dan tempat eksekusi berbeda
- *Resource* yang menangani setiap aktivitas berbeda
- *Message* setiap departemen ke aktivitas tidak saling bergantung

3.6 *Scheduling* dengan Sistem Manufaktur Fleksibel

Jika konsep paralelisasi diterapkan dengan *resource* yang sudah memadai untuk kondisi saat ini, maka akibatnya adalah terjadinya keterlambatan selesainya proses dan lamanya waktu tunggu jika jumlah *resource* tetap sama. Oleh karena itu, pada penelitian ini, *scheduling* dengan Sistem Manufaktur Fleksibel (FMS) dilakukan dengan metode *First Come First Serve* (FCFS) yang diterapkan pada kakas bantu LEKIN. Konsep FMS dibagi menjadi 2 kondisi yaitu:

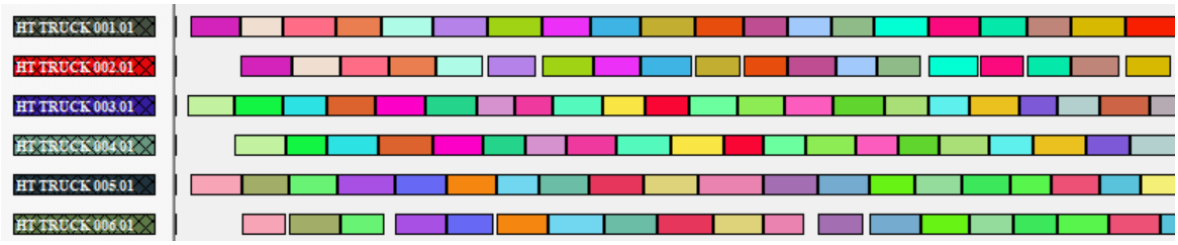
1) Satu mesin dapat berubah fungsi sehingga dapat menangani banyak job
2) Dua mesin dengan fungsi yang berbeda dimana salah satu mesin dapat berubah fungsi sehingga dapat apabila mesin lainnya mengalami kendala, mesin yang dapat berubah fungsi membantu mesin yang mengalami kendala tersebut

Langkah-langkah untuk *scheduling* pada tesis ini adalah:
Step 1. Menentukan jumlah *machine* dan *job* untuk dilakukan *scheduling*
Step 2. Menentukan durasi eksekusi atau *processing time* setiap *job*
Step 3. Dengan metode FCFS, *scheduling machine* dan *job* dilakukan pada kakas bantu LEKIN
Step 4. *Output* dari LEKIN adalah hasil *scheduling* yang memperlihatkan mesin dapat melakukan *job* secara kontinu terus menerus hingga seluruh *job* selesai diproses oleh mesin

- Jika *output* LEKIN adalah mesin dapat melakukan *job* secara kontinu terus menerus tanpa adanya *waiting time* terlalu lama maka mesin tersebut berjalan normal
- Jika *output* LEKIN adalah mesin melakukan *job* tetapi ditengah-tengah proses melakukan *job* terdapat *waiting time* yang nantinya akan mengakibatkan pertambahan durasi eksekusi dan molornya selesai aktivitas, maka fungsi mesin lainnya dapat membantu mesin yang bermasalah agar dapat menyelesaikan aktivitas tepat waktu bahkan lebih cepat dari yang seharusnya

Step 5. Dilakukan perhitungan waktu baru untuk mesin yang berubah fungsi dan percepatan waktu aktivitas setelah dikenai scheduling, dan rata-rata jumlah mesin berubah fungsi
Step 6. Output akhir adalah jumlah mesin berubah fungsi setiap hari dan rata-rata mesin berubah fungsi setiap bulan

Contoh output LEKIN dengan metode FCFS seperti pada Gambar 3.15. Gambar 3.15 menunjukkan bahwa untuk HT Truck 01, HT Truck 03, HT Truck 04, dan HT Truck 05 ke Departemen Karantina berjalan normal sehingga mesin lainnya tidak perlu berubah fungsi, sementara untuk HT Truck 02 dan HT Truck 06 terdapat *waiting time* sehingga proses tidak berlangsung kontinu yang mengakibatkan proses pada Departemen *Behandle* menjadi lama, maka pada kondisi seperti ini mesin lainnya yaitu RTGC berubah fungsi menjadi HT Truck untuk membantu mengantarkan *container* ke Departemen *Behandle*.



Gambar 3.15 Contoh *output scheduling* dengan LEKIN

3.7 Optimasi dengan *Goal Programming*

Pada penelitian tesis ini, *goal* atau sasaran yang ingin dicapai adalah meminimumkan waktu. *Goal* waktu diperoleh dari rata-rata setiap aktivitas per blok proses. Sedangkan yang ingin dioptimasi adalah waktu maksimum setiap aktivitas per departemen. Optimasi dilakukan per blok proses, yaitu *discharge, behandle*, karantina, dan *delivery*. Optimasi barang dan dokumen dilakukan terpisah karena menurut analisa dari kakas bantu Disco dan analisa bersama *expert*, yang membuat lama *dwelling time* sebuah *container* adalah karena waktu dokumen selesai lebih lama daripada waktu barang selesai seperti pada Tabel 3.10. Dari data pada Tabel 3.10 diketahui bahwa waktu rata-rata selesainya barang dari bulan Januari hingga Maret adalah 1,7 jam sedangkan untuk waktu rata-rata selesainya dokumen dari bulan Januari hingga Maret adalah 5,3 jam. Sementara, prinsip biaya pada optimasi ini adalah jika goal waktu berhasil dicapai, maka berapakah biaya yang bisa dihemat untuk setiap blok proses.

Tabel 3.10 Perbandingan waktu rata-rata barang dan dokumen PT TPS

Waktu rata-rata Barang (*Container)* (dalam jam)	Waktu rata-rata Dokumen (dalam jam)
1,654279	5,271168

Langkah-langkah optimasi waktu dan biaya dengan *Goal programming* adalah sebagai berikut:

1. Memisahkan aktivitas per blok proses

Discharge
Document_Entry_via_PDE
Vessel_Berthing_Process
Discharge_Container
Bring_Container_to_Yard
Stack_Container_in_Yard

Karantina
Verification_Document_Quarantine

Create_Job_Order_Document_Quarantine
Bring_Container_from_Yard_to_Quarantine
Stack_Container_in_Quarantine_Area
Check_Goods_Quarantine
Create_document_KH/KT
Send_Certificate_KH/KT_Info
Stack_Container_in_Yard_From_Quarantine

Behandle
Verification_Document_Behandle
Create_Job_Order_Document_Behandle
Stack_Container_in_Behandle_Area
Check_Goods_Behandle
Create_document_LHP
Bring_Container_from_Yard_to_Behandle
Stack_Container_in_Yard_From_Behandle
Create_document_SPPB

Delivery
Create_Job_Order_Document_Delivery
Truck_in
Dispatch_WQ_Delivery_to_CHE
Determine_Container_Type
Determining_Uncontainer/Determining_Dry/Determining_Reefer
Decide_Task_Before_Lift_Container
Prepare_Tools/Unplug_Reefer_Cable
Lift_on_Container_Truck
Truck_Go_To_Gate_Out
Check_Container_before_Truck_out
Truck_Out

2. Memisahkan aktivitas berdasarkan kategori barang dan dokumen

AKTIVITAS *DISCHARGE*

Dokumen
Document_Entry_via_PDE

Barang
Vessel_Berthing_Process
Discharge_Container

Bring_Container_to_Yard
Stack_Container_in_Yard

AKTIVITAS KARANTINA
Dokumen
Verification_Document_Quarantine
Create_Job_Order_Document_Quarantine
Create_document_KH/KT
Send_Certificate_KH/KT_Info
Barang
Bring_Container_from_Yard_to_Quarantine
Stack_Container_in_Quarantine_Area
Check_Goods_Quarantine
Stack_Container_in_Yard_From_Quarantine

AKTIVITAS *BEHANDLE*
Dokumen
Verification_Document_Behandle
Create_Job_Order_Document_Behandle
Create_document_LHP
Create_document_SPPB
Barang
Bring_Container_from_Yard_to_Behandle
Stack_Container_in_Behandle_Area
Check_Goods_Behandle
Stack_Container_in_Yard_From_Behandle

AKTIVITAS *DELIVERY*
Dokumen
Create_Job_Order_Document_Delivery
Barang
Truck_in
Dispatch_WQ_Delivery_to_CHE
Determine_Container_Type
Determining_Uncontainer
Determining_Dry

Determining_Reefer
Decide_Task_Before_Lift_Container
Prepare_Tools
Unplug_Reefer_Cable
Lift_on_Container_Truck
Truck_Go_To_Gate_Out
Check_Container_before_Truck_out
Truck_Out

3. Menghitung waktu rata-rata dan maksimum serta biaya rata-rata dan maksimum setiap aktivitas per blok proses

AKTIVITAS *DISCHARGE*	***AVG TIME***	***MAX TIME***	***AVG COST***	***MAX COST***
Dokumen				
Document_Entry_via_PDE	5268,59	5345,07	0	0
Barang				
Vessel_Berthing_Process	50594,19	51130,39	54330,79	56815,03
Discharge_Container	438,71	445,95	502,81	528,66
Bring_Container_to_Yard	767,58	841,22	61,70	65,06
Stack_Container_in_Yard	438,89	444,58	23,35	24,31

AKTIVITAS KARANTINA	***AVG TIME***	***MAX TIME***	***AVG COST***	***MAX COST***
Dokumen				
Verification_Document_Quarantine	2242,02	3629,10	177,33	192,55
Create_Job_Order_Document_Quarantine	1216,84	1998,67	177,13	192,34
Create_document_KH/KT	2995,78	6179,00	413,15	432,25
Send_Certificate_KH/KT_Info	87,8369	91,2264	11,82	12,56
Barang				
Bring_Container_from_Yard_to_Quarantine	2932,98	5158,48	177,16	188,25
Stack_Container_in_Quarantine_Area	71,5399	169,897	177,08	188,65
Check_Goods_Quarantine	1542,71	2580,85	354,89	381,37
Stack_Container_in_Yard_From_Quarantine	1588,78	4754,66	177,34	188,61

AKTIVITAS *BEHANDLE*	AVG TIME	MAX TIME	AVG COST	MAX TIME
Dokumen				
Verification_Document_Behandle	245,63	249,89	6,77	6,83
Create_Job_Order_Document_Behandle	107667,73	182555,2	465,53	528,37
Create_document_LHP	3520,58	6283,42	22445,19	25975,81
Create_document_SPPB	23193,20	39855,23	2656,99	2679,45
Barang				
Bring_Container_from_Yard_to_Behandle	8156,97	13816,80	465,75	543,98
Stack_Container_in_Behandle_Area	8260,85	15681,06	468,89	519,17
Check_Goods_Behandle	1868,75	3610,46	1104,63	1233,14
Stack_Container_in_Yard_From_Behandle	83,75	187,83	459,67	510,91

AKTIVITAS *DELIVERY*	AVG TIME	MAX TIME	AVG COST	MAX COST
Barang				
Create_Job_Order_Document_Delivery	24435,42	33958,31	40,5915514	45,89
Truck_in	68025,93	284646,3	13,5214821	15,41
Dispatch_WQ_Delivery_to_CHE	60,42	74,75	1,20428193	1,36
Determine_Container_Type	137,71	330,83	2,71	3,06
Determining_Uncontainer	98,21	180	18,31	23,37
Determining_Dry	106,63	176,58	18,04	19,01
Determining_Reefer	112,10	277,02	22,85	23,37
Decide_Task_Before_Lift_Container	71,49	143,98	2,71	3,06
Prepare_Tools	592,90	2003,67	54,28	92,36
Unplug_Reefer_Cable	312,72409	381,64	14,92	17,91
Lift_on_Container_Truck	117,24619	195,75	19,49	20,28
Truck_Go_To_Gate_Out	2363,9795	5469,81	2,71	3,04
Check_Container_before_Truck_out	67,602614	76,24	2,71	3,06
Truck_Out	72,344744	81,21	13,54	15,32

4. Formulasi fungsi tujuan

Untuk meminimalkan waktu setiap aktivitas per blok proses, fungsi tujuan dapat diformulasikan sebagai berikut :

Meminimumkan $Z = \sum_{i=1}^{m}(d11 + d12)$

Untuk $i = 1,2, \dots, m$ tujuan

5. Fungsi Kendala

Dari penelitian ini ada satu tujuan atau sasaran yang ingin dicapai untuk membantu pengambil keputusan dalam membuat perencanaan, sasaran ini meliputi :

1. Meminimalkan waktu eksekusi aktivitas per blok proses

Waktu *goal* atau sasaran diambil dari rata-rata waktu eksekusi setiap aktivitas per blok proses, sedangkan waktu yang ingin dioptimasi mengikuti goal adalah waktu maksimum tiap aktivitas per blok proses.

Biaya hasil optimasi adalah biaya yang diperoleh jika waktu bisa mencapai *goal.* Semakin ditekan waktu atau semakin kecil waktu eksekusi maka biaya juga akan semakin kecil dan dapat dihemat sebesar pengurangan maksimum biaya dengan biaya hasil optimasi.

6. Model matematika

1. Meminimalkan waktu eksekusi aktivitas per blok proses

$$Xi + d11 - d12 = Pi$$

Keterangan:

Xi : waktu maksimum

Pi : waktu rata-rata *(goal)*

$d11$: nilai penyimpangan di bawah Pi

$d12$: nilai penyimpangan di atas Pi

2. Memperoleh biaya aktivitas per blok proses mengikuti *goal* waktu

Fungsi tujuan:

$$Min\ Z = \sum_{i=1}^{m} (BiXi)$$

Keterangan:

Bi : biaya per blok proses

7. Penyelesaian optimal
   Hasil kombinasi variabel keputusan dari hasil optimisasi yang dilakukan dengan LINGO diperoleh waktu yang sesuai dengan sasaran atau *goal* serta biaya yang harus dikeluarkan jika waktu berhasil diminimumkan.

3.8 Output

3.8.1 Metode Pengujian dengan Perhitungan Nilai *Fitness*

Dalam literatur proses *mining*, tidak ada *framework* umum yang digunakan untuk mengevaluasi model proses yang ditemukan. Hanya dinyatakan bahwa untuk membandingkan dan mengevaluasi model proses yang ditemukan dapat menggunakan hasil model proses yang ditemukan dari algoritma-algoritma berbeda dan kualitas *fitness* dari model proses yang ditemukan digunakan untuk mengevaluasi sebagai keputusan akhir jika salah satu metode lebih baik daripada metode yang lain. Sebuah model proses dengan *fitness* yang baik menunjukkan "kecocokan" dari model proses yang ditemukan dengan "realitas". Sebuah model proses ini disebut memiliki fitness sempurna jika semua traces di data log dapat diwakili oleh model proses dari awal sampai akhir. Tetapi jika banyak traces di data log tidak dapat diwakili oleh model model dari awal sampai akhir, maka model proses ini disebut memiliki *fitness* yang buruk. Formula di bawah ini digunakan untuk menghitung nilai fitness.

$$Fitness = 40\% * \frac{APA}{NOAAEL} + 60\% * \frac{APCLT}{NOTAEL}$$

Keterangan:

$NOAAEL$: $jumlah\ aktivitas\ di\ data\ log$

$NOTAEL$: $jumlah\ trace\ di\ data\ log$

APA : $semua\ aktivitas\ yang\ tercakup\ di\ model\ proses$

$APCLT$: $trace\ yang\ tercakup\ di\ model\ proses$

3.8.2 Perbandingan model proses dengan algoritma *Alpha Miner*

Untuk membandingkan hasil dari algoritma *Modified Time-based Alpha Miner* dan algoritma *Alpha*, maka perbandingan model proses, perbandingan jumlah *trace* dan pendefinisian *gateway* paralel dilakukan. Kakas bantu ProM digunakan untuk menemukan model proses dengan algoritma *Alpha*.

BAB IV

HASIL PENELITIAN DAN PEMBAHASAN

4.1 Hasil Penelitian

4.1.1 Lingkungan Uji Coba

Lingkungan uji coba meliputi perangkat keras dan perangkat lunak yang dijelaskan sebagai berikut:

1. Perangkat keras
   a. Prosesor: Intel® Core™ i7 CPU @ 3.1GHz
   b. Memori (RAM): 8 GB
   c. Tipe sistem: 64-bit sistem operasi
2. Perangkat lunak
   a. Sistem operasi: Windows 10
   b. Perangkat pengembang:
      - Java
      - Python
      - Microsoft excel
      - Kakas bantu Eclipse
      - Kakas bantu Python
      - Kakas bantu LEKIN
      - Kakas bantu Lingo
      - Kakas bantu WoPeD

4.2 *Proses Discovery* dengan *Modified Time-based Alpha Miner*

Data log yang digunakan pada eksperimen tesis ini untuk modifikasi algoritma yang diusulkan ada 3 jenis data log. Data log pertama berisi 100 *case*, data log kedua berisi 100 *case* dan data log ketiga adalah data log dari PT TPS hasil transfomasi yang telah dijelaskan pada Subbab 3.3.

Mengikuti langkah-langkah pengerjaan yang telah dijelaskan pada Subbab 3.4, berikut adalah hasil modifikasi algoritma Alpha dalam menemukan model proses dari data log pada Tabel 4.1.

Step 1. Seluruh relasi *sequence* (>) setiap aktivitas dari seluruh *case* di data log

Case ID001: A->B, B->D, C->E, E->F, D->F

Case ID002: A->B, B->D, C->E, E->F, D->F

Case ID003: A->B, B->D, C->E, E->F, D->F

Case ID004: A->B, B->D, C->E, E->F, D->F

.....

.....

Case ID100: A->C, C->D, E->F, D->F

Tabel 4.1 Potongan data log uji coba 1

Case	Aktivitas	Start time	End time	Ex Dur
ID001	A	1/28/15 10.05	1/28/15 10.07	0.02
ID001	B	1/28/15 10.10	1/28/15 10.22	0.12
ID001	C	1/28/15 10.15	1/28/15 10.39	0.24
ID001	D	1/28/15 10.27	1/28/15 11.02	0.34
ID001	E	1/28/15 10.41	1/28/15 10.53	0.12
ID001	F	1/28/15 11.07	1/28/15 11.28	0.21
ID002	A	1/28/15 11.33	1/28/15 11.36	0.03
ID002	B	1/28/15 11.40	1/28/15 11.54	0.14
ID002	C	1/28/15 11.44	1/28/15 12.14	0.29
ID002	D	1/28/15 11.59	1/28/15 12.28	0.29
ID002	E	1/28/15 12.22	1/28/15 12.34	0.12
ID002	F	1/28/15 12.44	1/28/15 12.56	0.11
ID003	A	1/28/15 13.01	1/28/15 13.10	0.09
ID003	B	1/28/15 13.15	1/28/15 13.28	0.12
ID003	C	1/28/15 13.21	1/28/15 13.36	0.15
ID003	D	1/28/15 13.30	1/28/15 13.43	0.13
ID003	E	1/28/15 13.38	1/28/15 13.53	0.14
ID003	F	1/28/15 13.59	1/28/15 14.12	0.13
ID004	A	1/28/15 14.24	1/28/15 14.29	0.04
ID004	B	1/28/15 14.31	1/28/15 14.39	0.08
ID004	C	1/28/15 14.35	1/28/15 14.49	0.13
ID004	D	1/28/15 14.44	1/28/15 14.53	0.09

Step 2. Seluruh relasi paralel (||) setiap aktivitas dari seluruh *case* di data log

Case ID001: B||C, D||E

Case ID002: B||C, D||E

Case ID003: B||C, D||E

Case ID004: B||C, D||E

.....

.....

Case ID100: D||E

Step 3. Hapus relasi *sequence* dan paralel yang duplikat untuk mendapatkan jumlah *trace* di data log

Step 4. Mendapatkan semua *trace* dengan relasi paralel dan *sequence* dari data log

Trace 1	A->B, B||C, B->D, D||E, C->E, E->F, D->F
Trace 2	A->C, C->D, D||E, E->F, D->F

Step 5. Menentukan transisi (T_L) dari *workflow net*

Transisi: A, B, C, D, E, F

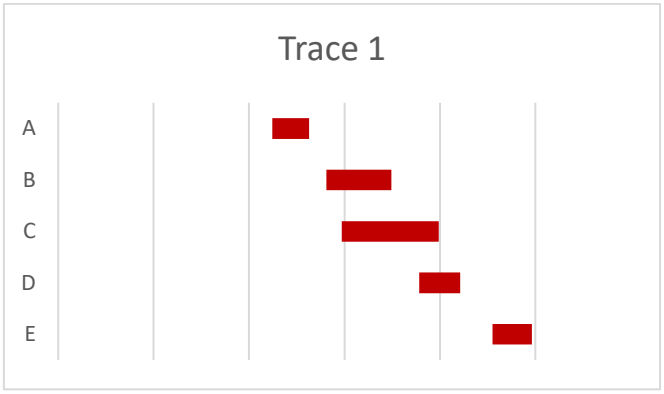
Step 6. Menentukan transisi *input* dari *workflow net*

Input: A

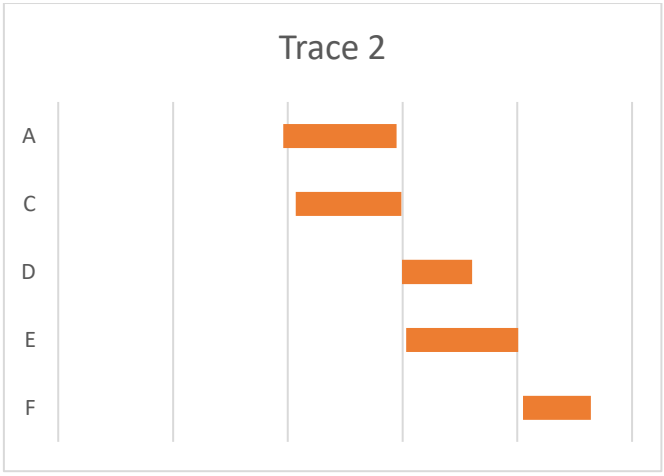
Step 7. Menentukan transisi *output* dari *workflow net*

Output: F

Step 8. Membuat *gantt chart* untuk semua *trace* berdasarkan *temporal causal relation* dan *control flow* untuk menunjukkan hubungan antara semua aktivitas dalam *trace* secara jelas



Gambar 4.1 *Gantt chart trace* 1

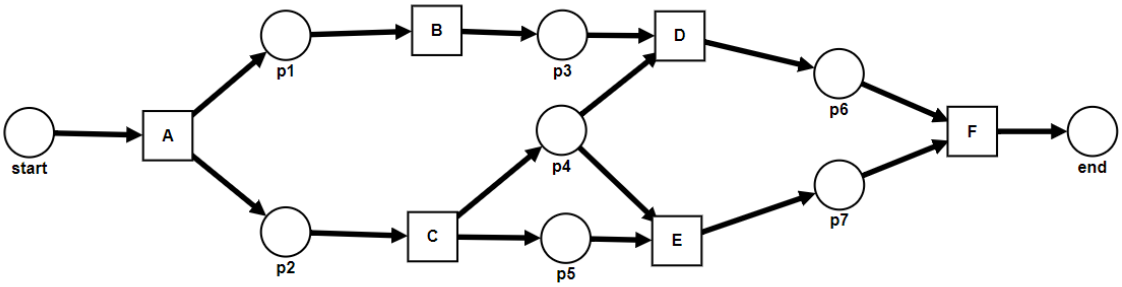


Gambar 4.2 *Gantt chart trace* 2

Step 9. Membuat *place*

Place: start(p0), p1, p2, p3, p4, p5, p6, p7, end(p8)

Step10. Menemukan model proses berdasarkan relasi *sequence* dan paralel



Gambar 4.3 Model Proses berdasarkan relasi *sequence* dan paralel data uji coba 1

Step11. Klasifikasikan kemungkinan tipe relasi paralel (AND, OR, XOR)

Relasi Aktivitas	Jumlah
A->B	80
B||C	80
A->C	20
B->D	80
C->D	20
D||E	100
C->E	80
E->F	100
D->F	100

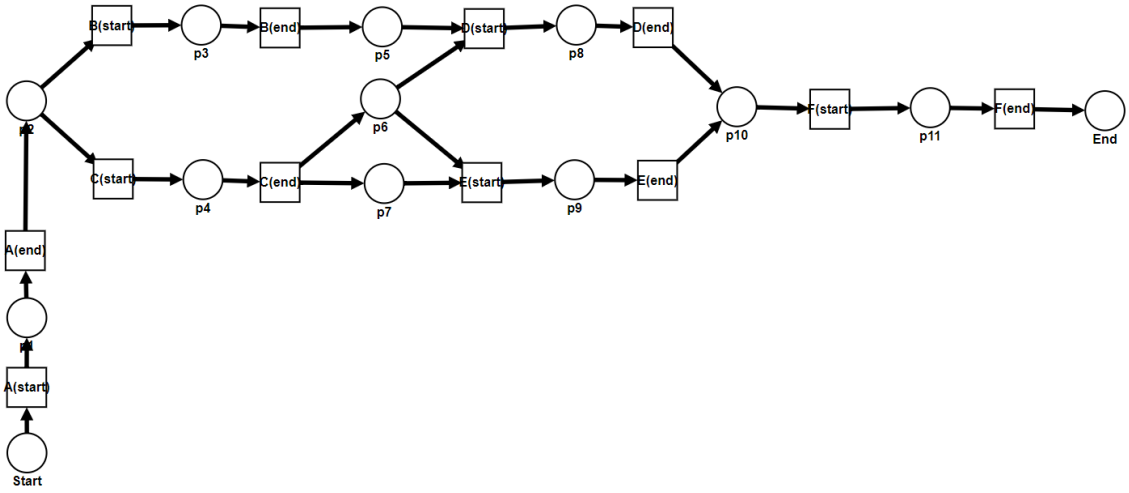
Aktivitas	*Min all sequence*	*Avg* paralel	*Avg all sequence*	Relasi
B||C	20	60	68,5714286	OR
D||E		80		AND

Tabel 4.2 Relasi model proses

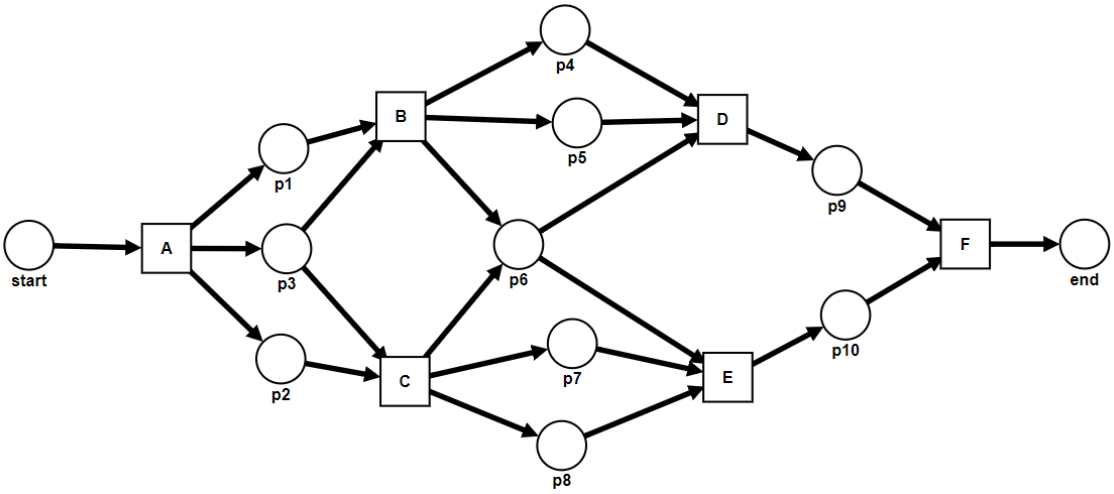
Input	*Split / Join*	*Output*
{Start}	*Sequence*	A
A	*OR Split*	B,C
B	*Sequence*	D
C	*Sequence*	D
C	*Sequence*	E
B,C	*OR Join*	D,E
C	*AND Split*	D,E
D,E	*AND Join*	F

F	*Sequence*	*{End}*

Step12. Model proses dalam bentuk Petri Net selesai



Gambar 4.4 Model proses dengan mengikut sertakan *start time* dan *end time* data log uji coba 1



Gambar 4.5 Model proses yang dengan salah satu informasi waktu data log uji coba 1

Pada Gambar 4.4, penggambaran model proses dengan mengikut sertakan tipe *start* dan *end*, sedangkan pada Gambar 4.5, model proses digambarkan hanya dengan salah satu tipe *start* atau *end* saja.

Selanjutnya adalah menerapkan algoritma modifikasi ke data log uji coba 2. Langkah yang sama dilakukan untuk menemukan model proses data log uji coba 2. Hasilnya adalah sebagai berikut:

Trace 1	A->B, B||C, C->D, D||E, B->D, C->E, D->F, E->F
Trace 2	A->C, C||B, B->D, D||E, C->D, B->E, D->F, E->F

Trace 3	A->B, B||C, C->E, E||D, B->E, C->D, E->F, D->F
Trace 4	A->C, C->D, D||E, C->E, D->F, E->F
Trace 5	A->C, C||B, B->D, E||D, B->E, C->D, E->F, D->F

Penentuan relasi paralel untuk data log uji coba 2. Langkah yang digunakan sama dengan data log uji coba 1 yaitu menghitung jumlah relasi *sequence* dan paralel dari seluruh *case* di data log.

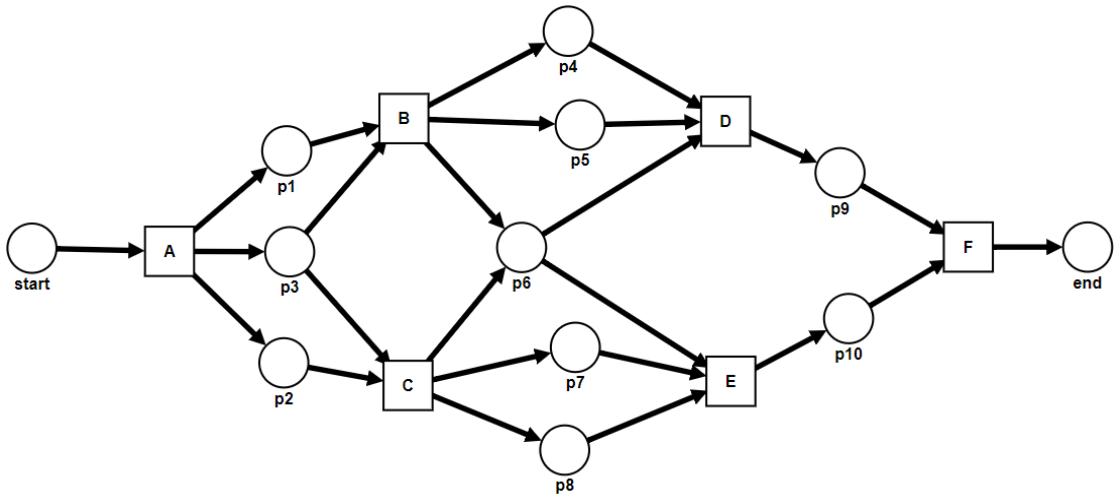
RELASI	JUMLAH
A->B	51
B||C	51
C->D	100
D||E	67
B->D	67
C->E	67
D->F	100
E->F	100
A->C	49
C||B	33
B->E	50
E||D	33

Aktivitas	*min all seq*	*avg PM (direct-indirect)*	*Avg all seq*	Relasi
B||C, C||B	49	58,5	73	OR
D||E, E||D		73		AND

Tabel 4.3 Relasi sequence dan paralel untuk data log uji coba 2

Input	***Split / Join***	***Output***
{Start}	*Sequence*	A
A	*OR Split*	B,C
B	*Sequence*	D
B	*Sequence*	E
C	*Sequence*	D
C	*Sequence*	E
B,C	*OR Join*	D,E
B,C	*AND Split*	D,E
D,E	*AND Join*	F
F	*Sequence*	*{End}*

Gambar 4.6 menunjukkan model proses yang ditemukan dengan algoritma modifikasi dengan *gateway* paralel OR dan AND.



Gambar 4.6 Model proses yang dengan salah satu informasi waktu data log uji coba 2

Langkah yang sama dilakukan untuk menemukan model proses data log PT TPS. Hasilnya adalah sebagai berikut:

Tabel 4.4 *Trace* yang terdapat pada data log PT TPS tanpa anomali

Trace 1 (Q-Hijau-Dry)	***Trace 2 (Q-Merah-Dry)***	***Trace 3 (Q-Hijau-Reefer)***	***Trace 4 (Q-Merah-Reefer)***
Entry Document via PDE	*Entry Document via PDE*	*Entry Document via PDE*	*Entry Document via PDE*
Vessel Berthing Process	*Vessel Berthing Process*	*Vessel Berthing Process*	*Vessel Berthing Process*
Discharge Container	*Discharge Container*	*Discharge Container*	*Discharge Container*
Bring Container to Yard	*Bring Container to Yard*	*Bring Container to Yard*	*Bring Container to Yard*
Stack Container in Yard	*Stack Container in Yard*	*Stack Container in Yard*	*Stack Container in Yard*
Verification Document Quarantine	*Verification Document Quarantine*	*Verification Document Quarantine*	*Verification Document Quarantine*
Create Job Order Document Quarantine	*Create Job Order Document Quarantine*	*Create Job Order Document Quarantine*	*Create Job Order Document Quarantine*
Bring Container from Yard to Quarantine	*Bring Container from Yard to Quarantine*	*Bring Container from Yard to Quarantine*	*Bring Container from Yard to Quarantine*
Stack Container in Quarantine Area	*Stack Container in Quarantine Area*	*Stack Container in Quarantine Area*	*Stack Container in Quarantine Area*

Check Goods Quarantine	*Check Goods Quarantine*	*Check Goods Quarantine*	*Check Goods Quarantine*
Create document KH/KT	*Create document KH/KT*	*Create document KH/KT*	*Create document KH/KT*
SendCertificateKH/KTInfo	*SendCertificateKH/KTInfo*	*SendCertificateKH/KTInfo*	*SendCertificateKH/KTInfo*
Stack Container in Yard From Quarantine	*Stack Container in Yard From Quarantine*	*Stack Container in Yard From Quarantine*	*Stack Container in Yard From Quarantine*
Verification Document Behandle	*Verification Document Behandle*	*Verification Document Behandle*	*Verification Document Behandle*
Create document SPPB	*Create Job Order Document Behandle*	*Create document SPPB*	*Create Job Order Document Behandle*
Create Job Order Document Delivery	*Bring Container from Yard to Behandle*	*Create Job Order Document Delivery*	*Bring Container from Yard to Behandle*
Truck in	*Stack Container in Behandle Area*	*Truck in*	*Stack Container in Behandle Area*
Dispatch WQ Delivery to CHE	*Check Goods Behandle*	*Dispatch WQ Delivery to CHE*	*Check Goods Behandle*
Determine Container Type	*Create document LHP*	*Determine Container Type*	*Create document LHP*
Determining Dry	*Stack Container in Yard From Behandle*	*Determining Reefer*	*Stack Container in Yard From Behandle*
Decide Task Before Lift Container	*Create document SPPB*	*Decide Task Before Lift Container*	*Create document SPPB*
Prepare Tools	*Create Job Order Document Delivery*	*Unplug Reefer cable*	*Create Job Order Document Delivery*
Lift on Container Truck	*Truck in*	*Lift on Container Truck*	*Truck in*
Truck Go To Gate Out	*Dispatch WQ Delivery to CHE*	*Truck Go To Gate Out*	*Dispatch WQ Delivery to CHE*
Check Container before Truck out	*Determine Container Type*	*Check Container before Truck out*	*Determine Container Type*
Truck Out	*Determining Dry*	*Truck Out*	*Determining Reefer*
	Decide Task Before Lift Container		*Decide Task Before Lift Container*

Prepare Tools	*Unplug Reefer cable*
Lift on Container Truck	*Lift on Container Truck*
Truck Go To Gate Out	*Truck Go To Gate Out*
Check Container before Truck out	*Check Container before Truck out*
Truck Out	*Truck Out*

Trace 5 (Q-Hijau-Uncontainer)	***Trace 6 (Q-Merah-Uncontainer)***	***Trace 7 (Hijau-Dry)***	***Trace 8 (Merah-Dry)***
Entry Document via PDE	*Entry Document via PDE*	*Entry Document via PDE*	*Entry Document via PDE*
Vessel Berthing Process	*Vessel Berthing Process*	*Vessel Berthing Process*	*Vessel Berthing Process*
Discharge Container	*Discharge Container*	*Discharge Container*	*Discharge Container*
Bring Container to Yard	*Bring Container to Yard*	*Bring Container to Yard*	*Bring Container to Yard*
Stack Container in Yard	*Stack Container in Yard*	*Stack Container in Yard*	*Stack Container in Yard*
Verification Document Quarantine	*Verification Document Quarantine*	*Verification Document Behandle*	*Verification Document Behandle*
Create Job Order Document Quarantine	*Create Job Order Document Quarantine*	*Create document SPPB*	*Create Job Order Document Behandle*
Bring Container from Yard to Quarantine	*Bring Container from Yard to Quarantine*	*Create Job Order Document Delivery*	*Bring Container from Yard to Behandle*
Stack Container in Quarantine Area	*Stack Container in Quarantine Area*	*Truck in*	*Stack Container in Behandle Area*
Check Goods Quarantine	*Check Goods Quarantine*	*Dispatch WQ Delivery to CHE*	*Check Goods Behandle*

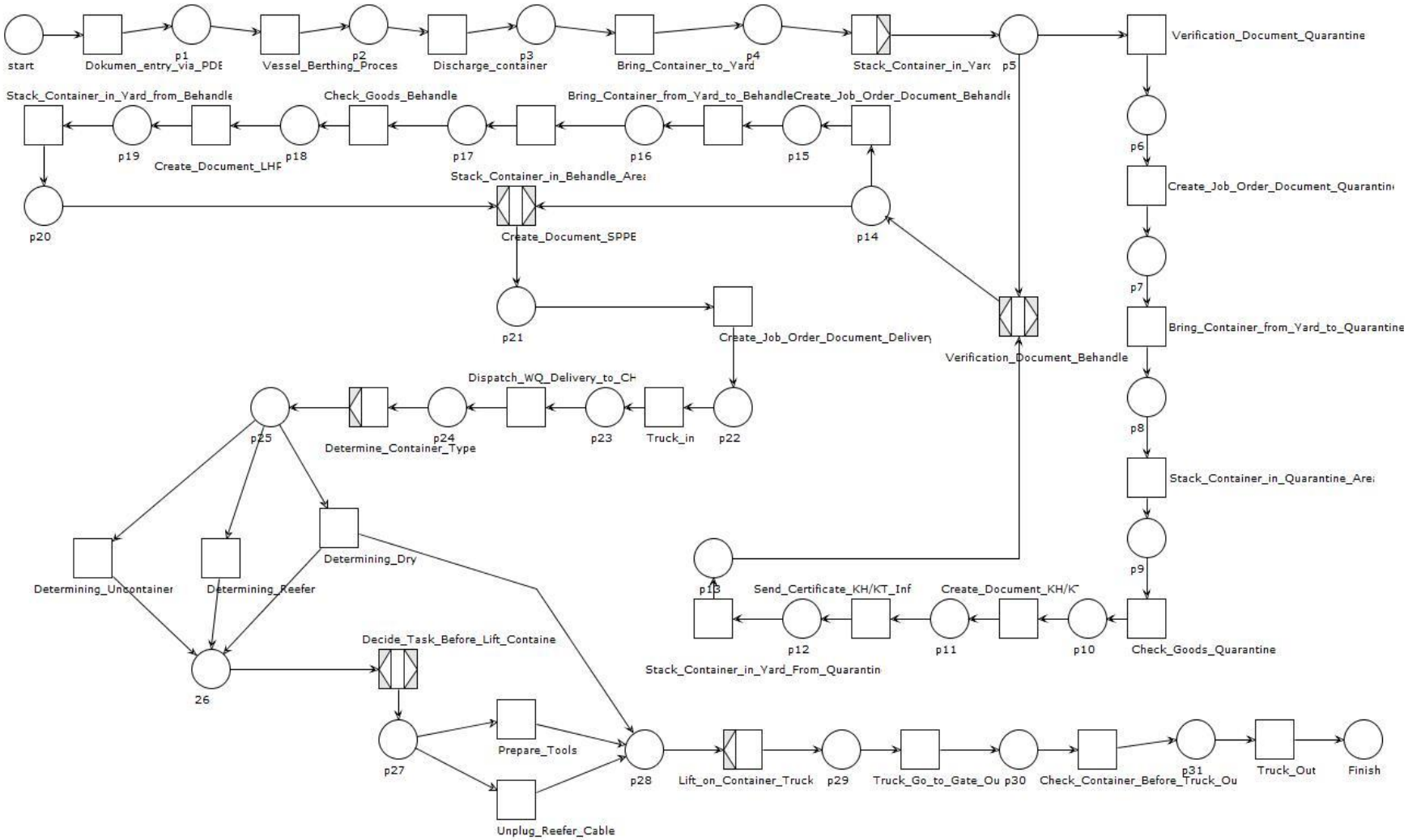
Create document KH/KT	*Create document KH/KT*	*Determine Container Type*	*Create document LHP*
SendCertificateKH/KTInfo	*SendCertificateKH/KTInfo*	*Determining Dry*	*Stack Container in Yard From Behandle*
Stack Container in Yard From Quarantine	*Stack Container in Yard From Quarantine*	*Decide Task Before Lift Container*	*Create document SPPB*
Verification Document Behandle	*Verification Document Behandle*	*Prepare Tools*	*Create Job Order Document Delivery*
Create document SPPB	*Create Job Order Document Behandle*	*Lift on Container Truck*	*Truck in*
Create Job Order Document Delivery	*Bring Container from Yard to Behandle*	*Truck Go To Gate Out*	*Dispatch WQ Delivery to CHE*
Truck in	*Stack Container in Behandle Area*	*Check Container before Truck out*	*Determine Container Type*
Dispatch WQ Delivery to CHE	*Check Goods Behandle*	*Truck Out*	*Determining Dry*
Determine Container Type	*Create document LHP*		*Decide Task Before Lift Container*
Determining Uncontainer	*Stack Container in Yard From Behandle*		*Prepare Tools*
Decide Task Before Lift Container	*Create document SPPB*		*Lift on Container Truck*
Prepare Tools	*Create Job Order Document Delivery*		*Truck Go To Gate Out*
Lift on Container Truck	*Truck in*		*Check Container before Truck out*
Truck Go To Gate Out	*Dispatch WQ Delivery to CHE*		*Truck Out*
Check Container before Truck out	*Determine Container Type*		
Truck Out	*Determining Uncontainer*		

Decide Task Before Lift Container
Prepare Tools
Lift on Container Truck
Truck Go To Gate Out
Check Container before Truck out
Truck Out

Trace 9 (Hijau-Reefer)	***Trace 10 (Merah-reefer)***	***Trace 11 (Hijau-Uncontainer)***	***Trace 12 (merah-Uncontainer)***
Entry Document via PDE	*Entry Document via PDE*	*Entry Document via PDE*	*Entry Document via PDE*
Vessel Berthing Process	*Vessel Berthing Process*	*Vessel Berthing Process*	*Vessel Berthing Process*
Discharge Container	*Discharge Container*	*Discharge Container*	*Discharge Container*
Bring Container to Yard	*Bring Container to Yard*	*Bring Container to Yard*	*Bring Container to Yard*
Stack Container in Yard	*Stack Container in Yard*	*Stack Container in Yard*	*Stack Container in Yard*
Verification Document Behandle	*Verification Document Behandle*	*Verification Document Behandle*	*Verification Document Behandle*
Create document SPPB	*Create Job Order Document Behandle*	*Create document SPPB*	*Create Job Order Document Behandle*
Create Job Order Document Delivery	*Bring Container from Yard to Behandle*	*Create Job Order Document Delivery*	*Bring Container from Yard to Behandle*
Truck in	*Stack Container in Behandle Area*	*Truck in*	*Stack Container in Behandle Area*
Dispatch WQ Delivery to CHE	*Check Goods Behandle*	*Dispatch WQ Delivery to CHE*	*Check Goods Behandle*
Determine Container Type	*Create document LHP*	*Determine Container Type*	*Create document LHP*
Determining Reefer	*Stack Container in Yard From Behandle*	*Determining Uncontainer*	*Stack Container in Yard From Behandle*

Decide Task Before Lift Container	*Create document SPPB*	*Decide Task Before Lift Container*	*Create document SPPB*
Unplug Reefer cable	*Create Job Order Document Delivery*	*Prepare Tools*	*Create Job Order Document Delivery*
Lift on Container Truck	*Truck in*	*Lift on Container Truck*	*Truck in*
Truck Go To Gate Out	*Dispatch WQ Delivery to CHE*	*Truck Go To Gate Out*	*Dispatch WQ Delivery to CHE*
Check Container before Truck out	*Determine Container Type*	*Check Container before Truck out*	*Determine Container Type*
Truck Out	*Determining Reefer*	*Truck Out*	*Determining Uncontainer*
	Decide Task Before Lift Container		*Decide Task Before Lift Container*
	Unplug Reefer cable		*Unplug Reefer cable*
	Lift on Container Truck		*Lift on Container Truck*
	Truck Go To Gate Out		*Truck Go To Gate Out*
	Check Container before Truck out		*Check Container before Truck out*
	Truck Out		*Truck Out*

Dari keseluruhan trace yang diperoleh dari data log PT TPS, model proses yang ditemukan dengan menggunakan algoritma modifikasi Alpha ditunjukkan pada Gambar 4.7.



Gambar 4.7 Hasil dicovery dengan data log PT TPS

Relasi model proses dari data log PT TPS dijabarkan pada Tabel 4.5. Didapatkan bahwa seluruh relasi paralel pada model proses adalah XOR.

Tabel 4.5 Hasil relasi yang ditemukan dengan data log PT TPS

Input	**Relasi**	***Output***
{}	*Sequence*	*Document_entry_via_PDE*
Document_entry_via_PDE	*Sequence*	*Vessel_berthing_process*
Vessel_berthing_process	*Sequence*	*Discharge_container*
Discharge_container	*Sequence*	*Bring_container_to_yard*
Bring_container_to_yard	*Sequence*	*Stack_container_in_yard*
Stack_container_in_yard	*Xor split*	*Verification_document_quarantine*
Stack_container_in_yard	*Xor split*	*Verification_document_behandle*
Verification_document_quarantine	*Sequence*	*Create_job_order_document_quarantine*
Create_job_order_document_quarantine	*Sequence*	*Bring_container_from_yard_to_quarantine*
Bring_container_from_yard_to_quarantine	*Sequence*	*Stack_container_in_quarantine_area*
Stack_container_in_quarantine_area	*Sequence*	*Check_goods_quarantine*
Check_goods_quarantine	*Sequence*	*Create_document_KH/KT*
Create_document_KH/KT	*Sequence*	*Send_certificate_KH/KT_Info*
Send_certificate_KH/KT_Info	*Sequence*	*Stack_container_in_yard_from_quarantine*
Stack_container_in_yard_from_quarantine	*Xor join*	*Verification_document_behandle*
Stack_container_in_yard	*Xor join*	*Verification_document_behandle*
Verification_document_behandle	*Xor split*	*Create_document_sppb*
Verification_document_behandle	*Xor split*	*Create_job_order_document_behandle*
Create_job_order_document_behandle	*Sequence*	*Stack_container_in_behandle_area*
Stack_container_in_behandle_area	*Sequence*	*Check_goods_behandle*
Check_goods_behandle	*Sequence*	*Create_document_LHP*
Create_document_LHP	*Sequence*	*Bring_container_from_yard_to_behandle*
Bring_container_from_yard_to_behandle	*Sequence*	*Stack_container_in_yard_from_behandle*
Stack_container_in_yard_from_behandle	*Xor join*	*Create_document_sppb*
Verification_document_behandle	*Xor join*	*Create_document_sppb*
Create_document_sppb	*Sequence*	*Create_job_order_document_delivery*
Create_job_order_document_delivery	*Sequence*	*Truck_in*
Truck_in	*Sequence*	*Dispatch_WQ_delivery_to_CHE*
Dispatch_WQ_delivery_to_CHE	*Sequence*	*Determine_container_type*
Determine_container_type	*Xor split*	*Determining_dry*
Determine_container_type	*Xor split*	*Determining_reefer*
Determine_container_type	*Xor split*	*Determining_uncontainer*
Determining_dry	*Xor join*	*Decide_task_before_lift_container*
Determining_reefer	*Xor join*	*Decide_task_before_lift_container*
Determining_uncontainer	*Xor join*	*Decide_task_before_lift_container*
Decide_task_before_lift_container	*Xor split*	*Prepare_tools*
Decide_task_before_lift_container	*Xor split*	*Unplug_reefer_cable*
Decide_task_before_lift_container	*Xor split*	*Lift_on_container_truck*
Prepare_tools	*Xor join*	*Lift_on_container_truck*
Unplug_reefer_cable	*Xor join*	*Lift_on_container_truck*
Decide task before lift container	*Xor join*	*Lift_on_container_truck*
Lift_on_container_truck	*Sequence*	*Truck_go_to_gate_out*
Truck_go_to_gate_out	*Sequence*	*Check_container_before_truck_out*
Check_container_before_truck_out	*Sequence*	*Truck_out*
Truck_out	*Sequence*	*{}*

4.2.1 Analisis

4.2.1.1 Deskripsi Umum Sistem

Sistem yang dirancang ditujukan untuk menghasilkan model proses dari data log. Kegiatan menghasilkan model proses dari data log disebut *process discovery*. Sebelum menggunakan fungsi *process discovery*, pengguna memasukkan data berupa data log. Data log yang dimasukkan akan ditampilkan dalam sistem. Kemudian, pengguna memicu sistem untuk melakukan *process discovery*. Model proses yang dihasilkan dari *process dscovery* ditampilkan pada sistem secara langsung.

4.2.1.2 Kebutuhan Fungsional Sistem

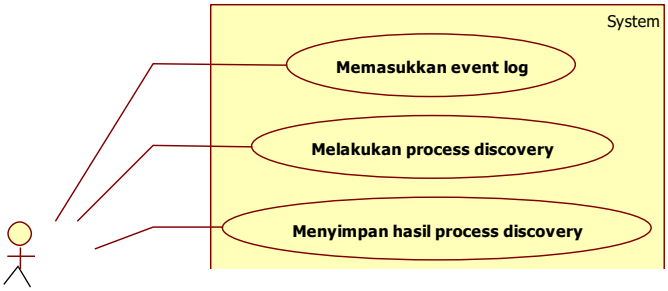
Kebutuhan fungsional berisi kebutuhan utama yang harus dipenuhi oleh sistem agar dapat bekerja dengan baik. Kebutuhan fungsional mendefinisikan layanan yang harus disediakan oleh sistem, bagaimana reaksi terhadap masukan, dan apakah yang harus dilakukan sistem pada situasi khusus. Daftar kebutuhan fungsional dapat dilihat pada .

Tabel 4.6 Daftar kebutuhan fungsional sistem

Kode Kebutuhan	Kebutuhan Fungsional	Deskripsi
KK-F001	Memasukkan data log	Pengguna memasukkan data berupa data log dalam format excel.
KK-F002	Melakukan *process discovery*	Pengguna memilih untuk melakukan *process discovery* pada data log yang dimasukkan sebelumnya.
KK-F003	Menyimpan hasil *process discovery*	Pengguna menyimpan model proses yang dihasilkan dari *process discovery*.

4.2.1.3 Kasus Penggunaan Sistem

Kasus-kasus penggunaan dalam sistem ini dijelaskan secara rinci pada bagian ini. Kasus penggunaan secara umum digambarkan oleh salah satu model UML, yaitu diagram kasus penggunaan. Rincian kasus penggunaan berisi spesifikasi kasus penggunaan, dan diagram aktivitas untuk masing-masing kasus penggunaan. Diagram kasus penggunaan dapat dilihat Gambar 4.8. Daftar kode diagram kasus penggunaan sistem dapat dilihat pada Tabel 4.7.



Gambar 4.8 Diagram Kasus Penggunaan Sistem

Tabel 4.7 Daftar Kode Diagram Kasus Penggunaan

Kode Kasus Penggunaan	Nama
KK-UC001	Memasukkan data log
KK-UC002	Melakukan *process discovery*
KK-UC003	Menyimpan hasil *process discovery*

4.2.2 Perancangan

Bagian ini membahas mengenai perancangan antarmuka pada sistem. Untuk melakukan *process discovery*, pertama pengguna harus memasukkan data log. Kemudian, memanggil fungsi *process discovery*, dan menampilkan model proses yang dihasilkan.

4.2.2.1 Fungsional Process Discovery

Lapisan fungsional *process discovery* berfungsi untuk melakukan *process discovery*. Secara umum, terdapat 3 step utama dalam *process discovery* yaitu memasukkan masukan data log, memanggil fungsi *process discovery*, dan menampilkan hasilnya. Hasil untuk sistem ini adalah semua *sequence* dan paralel dari data log serta *gateway* paralel dari model proses yang terbentuk.

Tabel 4.8 Program untuk melakukan proses *discovery* dengan *Modified Time-based Alpha*

```
package node;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Date;
import java.util.List;
import node.*;

public class MainProgram {

   public static void main(String[] args)
   {
                String csvFile = "C:\\EvlogAlphaModif.csv";
                String splitBy = ",";

      List<Combination> combinations = readCSV(csvFile,splitBy);
      int counter = 0;

      for(Combination comb : combinations)
      {
                //comb.show();
      }
      printWholeSequences(combinations);
```

```
      // Instantiate a Date object
      Date date = new Date();

      // display time and date using toString()
      System.out.println(date.toString());

   }

   protected static void printWholeSequences(List<Combination> combinations)
   {
      List<Sequence> newsequences = new ArrayList<Sequence>();
      List<Sequence> parallelSequences = getAllParallelSequences(combinations);

      for(Combination comb : combinations)
      {
         List<Sequence> sequences = comb.getSequences();
         for(Sequence seq : sequences) {
                  if(!inSequences(newsequences,seq) &&
!inSequences(parallelSequences,seq)) {
                           newsequences.add(seq);
                  }
         }
      }
      System.out.println(newsequences.size() + " SEQUENCES : ");
      Collections.sort(newsequences,Sequence.TaskComparator);
      for(Sequence s : newsequences)
      {
                  System.out.println(s);
      }
      System.out.println(parallelSequences.size() + " PARALEL SEQUENCES : ");
      for(Sequence s : parallelSequences)
      {
                           System.out.println(s.current.task +" || "+ s.next.task);
      }


   }

   protected static List<Sequence> getAllParallelSequences(List<Combination>
combinations)
   {
                  List<Sequence> parallelSequences = new ArrayList<Sequence>();
                  for(Combination comb : combinations)
      {
                           combineTwoSequences(parallelSequences,
comb.getParallelSequences());
      }
                  return parallelSequences;
   }
```

```
   protected static List<Sequence> combineTwoSequences(List<Sequence> one,
List<Sequence> two)
   {
                  for(Sequence s : two)
                  {
                        if(!inParallelSequences(one,s)) {
                        one.add(s);
                        }
                  }
                  return one;
   }

   protected static boolean inParallelSequences(List<Sequence>seqlist,Sequence seq)
   {
                  for(Sequence s : seqlist)
                  {
                        if(seq.current.equals(s.current) && seq.next.equals(s.next) ||
                              seq.next.equals(s.current) && seq.current.equals(s.next)){
                              return true;
                        }
                  }
                  return false;
   }

   protected static boolean inSequences(List<Sequence>seqlist,Sequence seq)
   {
                  for(Sequence s : seqlist)
                  {
                        if(seq.equals(s)) {
                              return true;
                        }
                  }
                  return false;
   }

   protected static List<Combination> readCSV(String csvLocation, String splitBy)
   {
                  Combination comb;
      int counter=1;

                  List<Combination> combinations = new ArrayList<Combination>();

      String line = "";
                  try (BufferedReader br = new BufferedReader(new
FileReader(csvLocation))) {
         while ((line = br.readLine()) != null) {

            // use comma as separator
            String[] split = line.split(splitBy);

            Node node = new Node(split[1],split[0],split[2],split[3]);
```

```
            int searchCombination = searchCombination(combinations,node.group);
            if(searchCombination > -1) {
                        comb = combinations.get(searchCombination);
            }else {
                        comb = new Combination(node.group);
               combinations.add(comb);
            }

            comb.addNode(node);
            counter++;
         }

         for(Combination c : combinations) {
                  c.closeCombination();
         }

         return combinations;
      }
      catch (IOException e) {
         e.printStackTrace();
         return null;
      }
   }

   protected static int searchCombination(List<Combination> combinations,String group)
   {
                  int counter = 0;
                  for(Combination cmb : combinations) {
                        if(cmb.groupName.equals(group)) {
                              return counter;
                        }
                        counter++;
                  }
                  return -1;
   }

}
```

Tabel 4.9 Program untuk menentukan *gateway* paralel dengan *Modified Time-based Alpha*

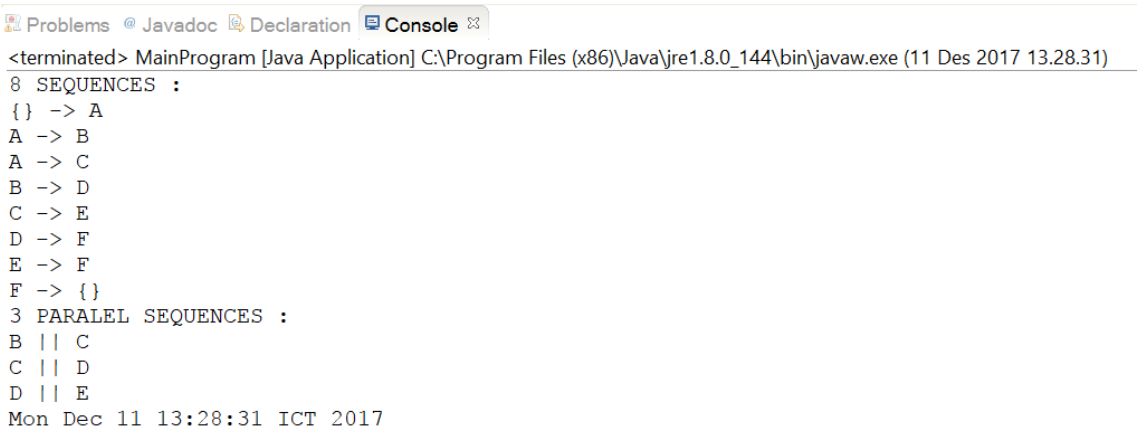
```
__author__ = 'yutikamelia'

import processing as pr
import copy

if __name__ == '__main__':
   file = " EvlogAlphaModif1.xlsx"
   excel = pr.excel()
   alpha = pr.alpha()
```

```
    rawData = excel.readData(file)
    dataDependency = alpha.getMatrixDependecy(copy.deepcopy(rawData))
    dataMeasure = alpha.getMatrixDependencyMeasure(copy.deepcopy(dataDependency))
    dataOverLap, dataCountOverLap = alpha.getOverlap(copy.deepcopy(rawData))
    dataMeasureOverlap =
alpha.multipleList2D(copy.deepcopy(dataOverLap),copy.deepcopy(dataMeasure)),
copy.deepcopy(dataMeasureOverlap))
    graph = alpha.getInitGraph(copy.deepcopy(rawData))
    graph = alpha.makeGraph(rawData,graph,dataMeasure,DT)
    graph =
alpha.getRelation(rawData,graph,dataDependency,dataCountOverLap,dataMeasureOverlap)
    graph = alpha.makeGraph(rawData,graph,data1Loop,1)
    for x in graph:
        print x,graph[x]
```

4.2.2.2 Halaman Proses Discovery

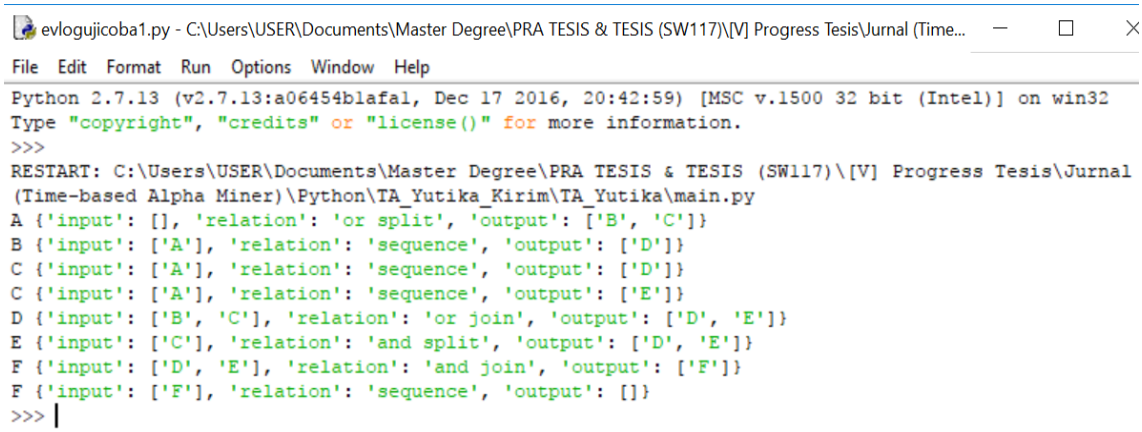
Halaman ini adalah halaman yang digunakan untuk melakukan *process discovery*. Untuk melakukan *process discovery*, pertama pengguna harus memasukkan data log. Kemudian, memanggil fungsi *process discovery*, dan menampilkan model proses yang dihasilkan.

Gambar 4.9 *Output* program dengan data log uji coba 1

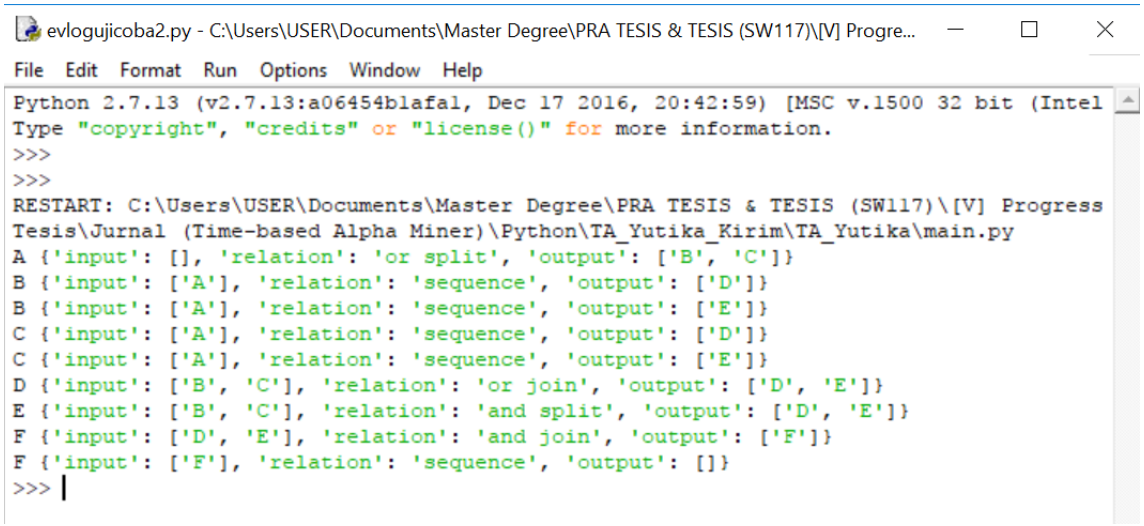


Gambar 4.10 *Output* program dengan data log uji coba 2

Gambar 4.11 *Output* program dengan data log PT TPS



Gambar 4.12 Relasi model proses data log uji coba 1



Gambar 4.13 Relasi model proses data log uji coba 2

Gambar 4.14 Relasi model proses data log PT TPS

4.3 Paralelisasi Aktivitas

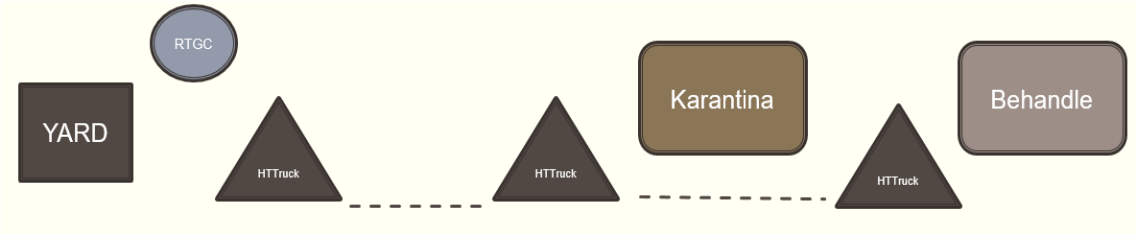
Dari aktivitas-aktivitas yang dijalankan pada PT TPS, terdapat Departemen Karantina dan *Behandle* yang dijalankan sekuential, dimana setelah *container* masuk ke departemen Karantina baru setelahnya masuk ke Departemen Behandle. Kedua departemen ini dapat dijalankan secara paralel, karena Departemen Karantina dan Behandle tidak saling bergantungan satu dengan yang lainnya, dengan alasan:

- Aktivitas setiap departemen independen
- Waktu dan tempat eksekusi berbeda
- *Resource* yang menangani setiap aktivitas berbeda
- *Message* setiap departemen ke aktivitas tidak saling bergantung

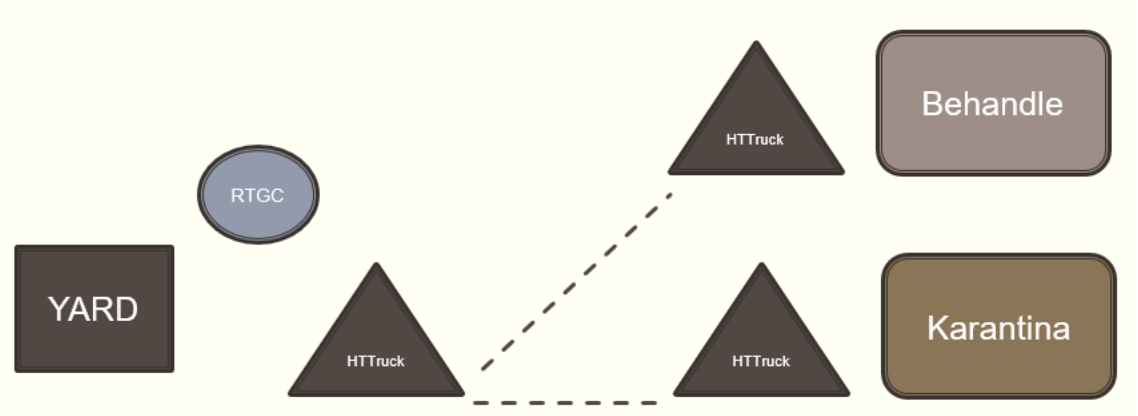
Setelah ketentuan paralelisasi diterapkan, maka Departemen Karantina dan Departemen *Behandle* ini dapat dieksekusi bersamaan. *Container* yang ditumpuk di *Yard* akan dapat dipindahkan ke Departemen Karantina dan *Behandle* secara bersamaan menurut bloknya sehingga dapat diharapkan dapat mengurangi *dwelling time.* Ilustrasi model proses PT TPS sebelum dan sesudah paralelisasi dijelaskan pada Gambar 4.15 dan Gambar 4.16.

Jika eksekusi pada kedua departemen ini dilakukan secara paralel, maka relasi model proses kedua departemen ini menjadi OR, karena Departemen Karantina dapat dijalankan atau tidak, sedangkan Departemen Behandle terbagi menjadi 2 yaitu Jalur Hijau dan Jalur Merah. Model proses PT TPS setelah paralelisasi ditunjukkan pada Gambar 4.17.

Batasan yang kami lakukan pada saat paralelisasi ini adalah *trace* atau jalur yang dapat dilakukan paralelisasi adalah jalur lengkap dimana kedua Departemen Karantina dan *Behandle* dieksekusi. Paralelisasi yang diterapkan masih pada keseluruhan departemen, tidak masuk ke aktivitas atau dokumen di dalam departemen.



Gambar 4.15 Ilustrasi model proses PT TPS sebelum paralelisasi



Gambar 4.16 Ilustrasi model proses PT TPS setelah paralelisasi

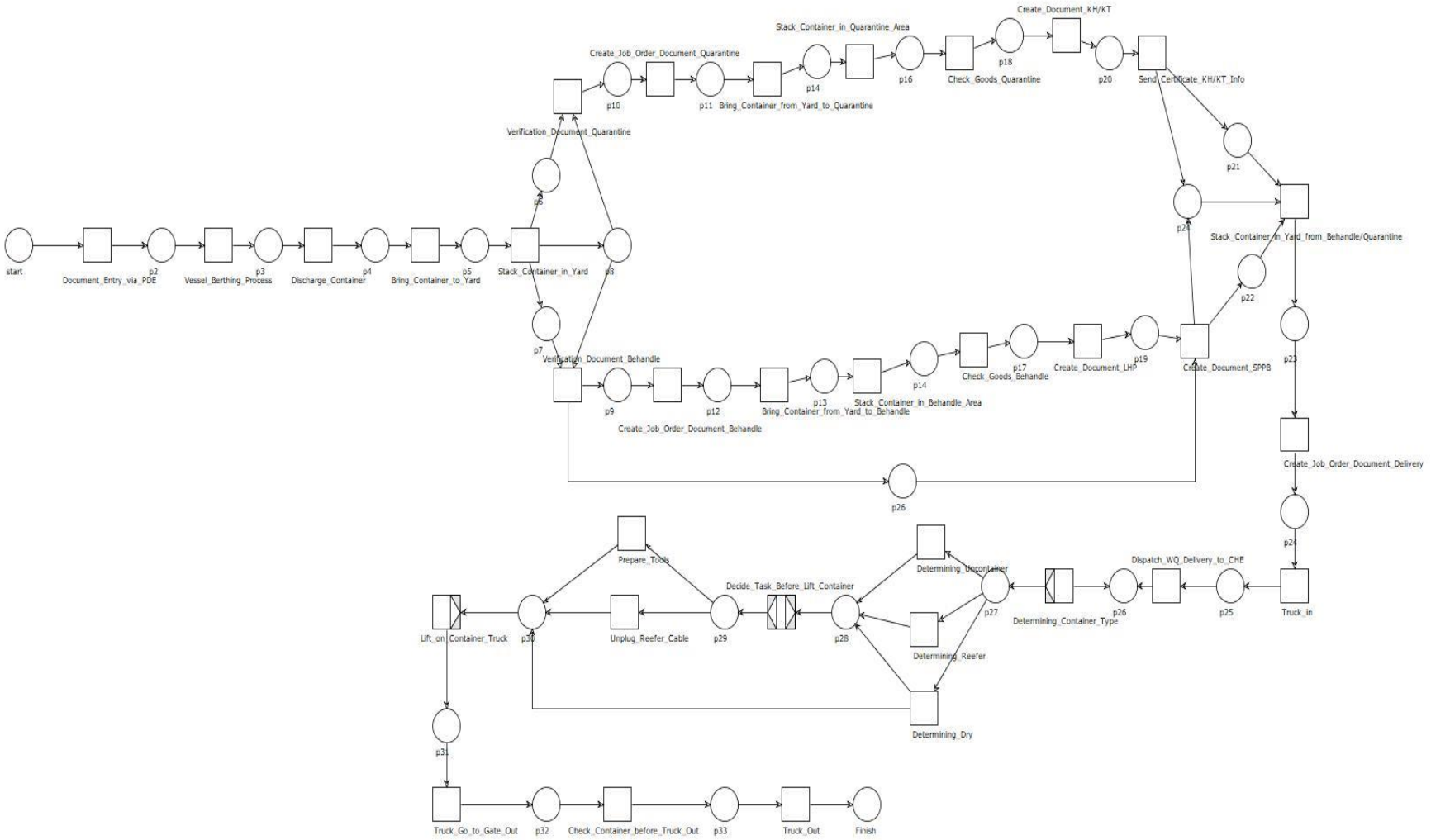
4.4 *Scheduling* dengan Sistem Manufaktur Fleksibel

Karena dilakukannya paralelisasi antara Departemen Karantina dan *Behandle*, maka munculah permasalah sebagai berikut:

- Kondisi saat ini yaitu setelah Departemen Karantina di eksekusi, maka *container* akan masuk ke Departemen *Behandle*.
- 33 RTGC dan 6 HT Truck yang dioperasikan pada kondisi saat ini berjalan lancar
- Karena Departemen Karantina dan *Behandle* pada kondisi yang diusulkan adalah paralel (dapat berjalan bersamaan sekaligus), maka 6 HT Truck yang berfungsi mengantarkan *container* ke Departemen Karantina dari *yard* mengalami kendala
- Maka *Scheduling* dengan Sistem Manufaktur Fleksibel dilakukan pada tahap ini dimana mesin RTGC dapat berfungsi tidak hanya memindahkan *container* dari yard ke HT Truck saja tetapi dapat membantu mengantarkan *container* ke Departemen Karantina dan Departemen *Behandle*
- Batasan yang dilakukan saat *scheduling*:
    - FMS terjadi pada mesin RTGC
    - Blok *container* (*dry, reefer dan uncontainer*) telah diketahui dan *container* telah disusun sesuai blok di *yard* dengan benar
    - Alur *flow container* dari awal sudah benar
    - Mesin yang digunakan 33 RTGC dan 6 HT Truck
    - Aktivitas yang terlibat adalah *Stack Container in Yard, Bring Container from Yard to Quarantine* dan *Bring Container from Yard to Behandle*

- Waktu *load* dan *unload* HT Truck setiap *container* adalah 5-7 menit. Waktu durasi perjalanan HT Truck diambil dari aktivitas *Bring Container from Yard to Quarantine* dan *Bring Container from Yard to Behandle*
- Waktu RTGC memindahkan *container* dari *Yard* ke HT Truck 3-5 menit
- Waktu RTGC untuk memindahkan *container* langsung adalah 15 menit
- *Job* adalah setiap container dengan durasi eksekusi masing-masing

Kakas bantu yang digunakan pada saat *Scheduling* adalah LEKIN dan metode yang digunakan untuk *Scheduling* adalah FCFS.



Gambar 4.17 Model proses PT TPS setelah paralelisasi

4.4.1 Penetuan Aktivitas dan Mesin yang digunakan saat *Scheduling*

Aktivitas dan mesin yang terlibat dalam *scheduling* dengan FMS dapat dilihat pada Tabel 4.10 dan Tabel 4.11.

Tabel 4.10 Aktivitas dan mesin departemen Karantina

Aktivitas	***MACHINE***
Stack_Container_in_Yard	RTGC
KARANTINA	***MACHINE***
Bring_Container_from_Yard_to_Quarantine	HT TRUCK
Stack_Container_in_Quarantine_Area	CC

Tabel 4.11 Aktivitas dan mesin departemen Behandle

Aktivitas	***MACHINE***
Stack_Container_in_Yard	RTGC
BEHANDLE	***MACHINE***
Bring_Container_from_Yard_to_Behandle	HT TRUCK
Stack_Container_in_Behandle_Area	CC

Jumlah mesin yang digunakan untuk *scheduling* adalah 33 RTGC dan 6 HT Truck.

HT Truck 01 dan HT Truck 02 digunakan untuk mengangkut *Container Dry.*

HT Truck 03 dan HT Truck 04 digunakan untuk mengangkut *Container Uncontainer.*

HT Truck 05 dan HT Truck 06 digunakan untuk mengangkut *Container Reefer.*

Simulasi *scheduling* dilakukan per hari selama satu bulan Februari dengan 100 *container* per harinya sebagai simulasi di kakas bantu LEKIN, dengan perbandingan *container dry, reefer* dan *uncontainer* 3:2:1. Pada Tabel 4.12 dijelaskan penjabaran *ID Container*, Tipe *container*, RTGC dan HT Truck untuk satu hari awal di bulan Februari.

Tabel 4.12 Data untuk *scheduling* dengan FMS

No	***ID Container***	**Tipe Container**	**RTGC**	**HT Truck**
1	ID001	*Dry*	01	001, 002
2	ID002	*Uncontainer*	02	003, 004
3	ID003	*Reefer*	03	005, 006
4	ID004	*Dry*	04	001, 002
5	ID005	*Uncontainer*	05	003, 004
6	ID006	*Reefer*	06	005, 006

7	ID007	*Dry*	07	001, 002
8	ID008	*Uncontainer*	08	003, 004
9	ID009	*Reefer*	09	005, 006
10	ID010	*Dry*	10	001, 002
11	ID011	*Uncontainer*	11	003, 004
12	ID012	*Reefer*	12	005, 006
13	ID013	*Dry*	13	001, 002
14	ID014	*Uncontainer*	14	003, 004
15	ID015	*Reefer*	15	005, 006
16	ID016	*Dry*	16	001, 002
17	ID017	*Uncontainer*	17	003, 004
18	ID018	*Reefer*	18	005, 006
19	ID019	*Dry*	19	001, 002
20	ID020	*Uncontainer*	20	003, 004
21	ID021	*Reefer*	21	005, 006
22	ID022	*Dry*	22	001, 002
23	ID023	*Uncontainer*	23	003, 004
24	ID024	*Reefer*	24	005, 006
25	ID025	*Dry*	25	001, 002
26	ID026	*Uncontainer*	26	003, 004
27	ID027	*Reefer*	27	005, 006
28	ID028	*Dry*	28	001, 002
29	ID029	*Uncontainer*	29	003, 004
30	ID030	*Reefer*	30	005, 006
31	ID031	*Dry*	31	001, 002
32	ID032	*Uncontainer*	32	003, 004
33	ID033	*Reefer*	33	005, 006
34	ID034	*Dry*	01	001, 002
35	ID035	*Uncontainer*	02	003, 004
36	ID036	*Reefer*	03	005, 006

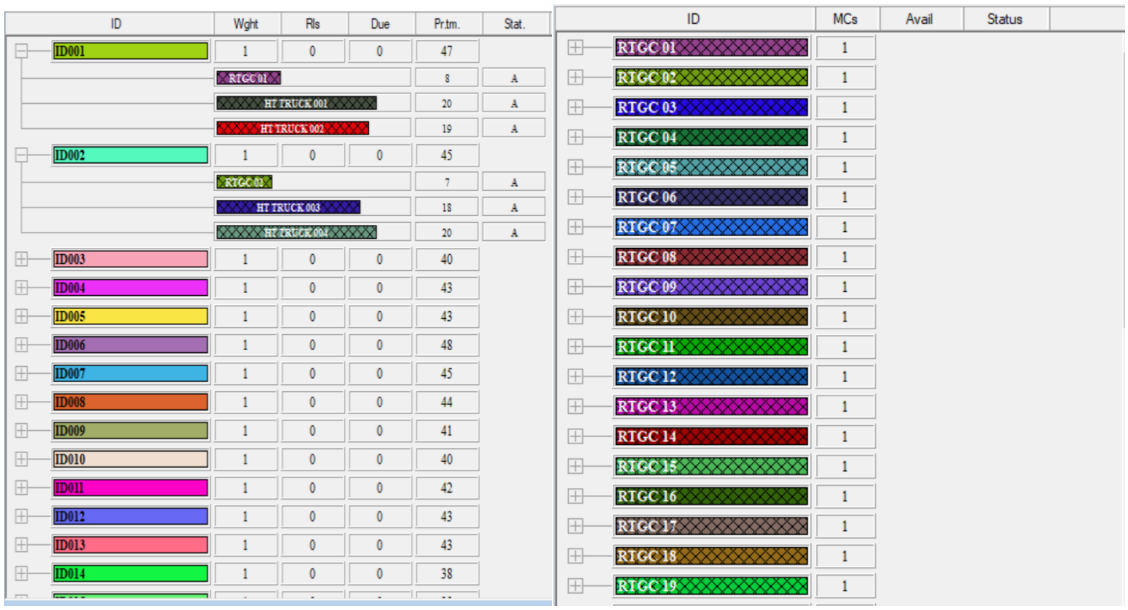
37	ID037	*Dry*	04	001, 002
38	ID038	*Uncontainer*	05	003, 004
39	ID039	*Reefer*	06	005, 006
40	ID040	*Dry*	07	001, 002
41	ID041	*Uncontainer*	08	003, 004
42	ID042	*Reefer*	09	005, 006
43	ID043	*Dry*	10	001, 002
44	ID044	*Uncontainer*	11	003, 004
45	ID045	*Reefer*	12	005, 006
46	ID046	*Dry*	13	001, 002
47	ID047	*Uncontainer*	14	003, 004
48	ID048	*Reefer*	15	005, 006
49	ID049	*Dry*	16	001, 002
50	ID050	*Uncontainer*	17	003, 004
51	ID051	*Reefer*	18	005, 006
52	ID052	*Dry*	19	001, 002
53	ID053	*Uncontainer*	20	003, 004
54	ID054	*Reefer*	21	005, 006
55	ID055	*Dry*	22	001, 002
56	ID056	*Uncontainer*	23	003, 004
57	ID057	*Reefer*	24	005, 006
58	ID058	*Dry*	25	001, 002
59	ID059	*Uncontainer*	26	003, 004
60	ID060	*Reefer*	27	005, 006
61	ID061	*Dry*	28	001, 002
62	ID062	*Uncontainer*	29	003, 004
63	ID063	*Reefer*	30	005, 006
64	ID064	*Dry*	31	001, 002
65	ID065	*Uncontainer*	32	003, 004
66	ID066	*Reefer*	33	005, 006

67	ID067	*Dry*	01	001, 002
68	ID068	*Uncontainer*	02	003, 004
69	ID069	*Reefer*	03	005, 006
70	ID070	*Dry*	04	001, 002
71	ID071	*Uncontainer*	05	003, 004
72	ID072	*Reefer*	06	005, 006
73	ID073	*Dry*	07	001, 002
74	ID074	*Uncontainer*	08	003, 004
75	ID075	*Reefer*	09	005, 006
76	ID076	*Dry*	10	001, 002
77	ID077	*Uncontainer*	11	003, 004
78	ID078	*Reefer*	12	005, 006
79	ID079	*Dry*	13	001, 002
80	ID080	*Uncontainer*	14	003, 004
81	ID081	*Reefer*	15	005, 006
82	ID082	*Dry*	16	001, 002
83	ID083	*Uncontainer*	17	003, 004
84	ID084	*Reefer*	18	005, 006
85	ID085	*Dry*	19	001, 002
86	ID086	*Uncontainer*	20	003, 004
87	ID087	*Reefer*	21	005, 006
88	ID088	*Dry*	22	001, 002
89	ID089	*Uncontainer*	23	003, 004
90	ID090	*Reefer*	24	005, 006
91	ID091	*Dry*	25	001, 002
92	ID092	*Uncontainer*	26	003, 004
93	ID093	*Reefer*	27	005, 006
94	ID094	*Dry*	28	001, 002
95	ID095	*Uncontainer*	29	003, 004
96	ID096	*Reefer*	30	005, 006

97	ID097	*Dry*	31	001, 002
98	ID098	*Uncontainer*	32	003, 004
99	ID099	*Reefer*	33	005, 006
100	ID100	*Dry*	01	001, 002

4.4.2 *Input* Mesin dan *Job* pada Kakas Bantu LEKIN

Inputan pada kakas bantu LEKIN adalah jumlah mesin, jumlah *job*, dan durasi eksekusinya. Pada penelitian ini, jumlah mesin adalah 33 RTGC dan 6 HT Truck. Sedangkan jumlah *job* adalah 100 sesuai jumlah *container* per harinya. Lalu masukkan durasi eksekusi setiap mesin yang mewakilkan durasi eksekusi setiap aktivitas yang dilakukan mesin ke dalam LEKIN sesuai ID Container. Input pada LEKIN ditunjukkan pada Gambar 4.18.

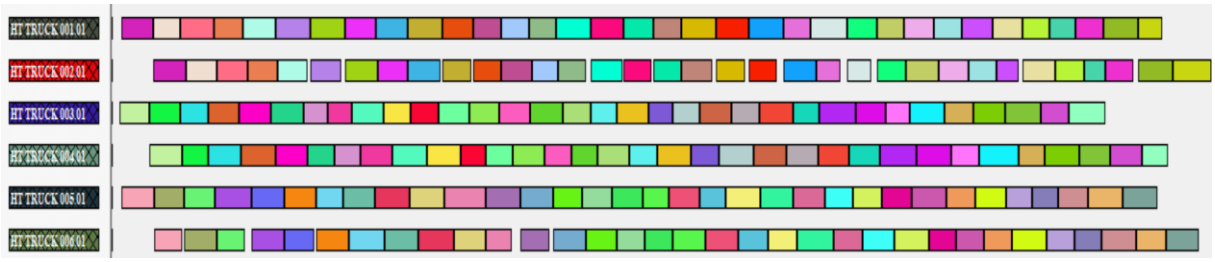


Gambar 4.18 *Inputan* mesin, *job* dan durasi pada LEKIN

4.4.3 *Scheduling* dengan Metode FCFS

Seperti yang telah dijelaskan pada subbab 2.2.5, salah satu metode untuk *scheduling* dengan FMS adalah menggunakan metode *First Come First Serve* (FCFS). Hasil *gantt chart* yang dihasilkan oleh LEKIN terlihat pada Gambar 4.19. Oleh karena itu, hasil scheduling berdasarkan data log PT TPS untuk HT Truck akan dijelaskan pada Tabel 4.13 untuk container Dry dan Karantina, Tabel 4.14 untuk container dry dan Behandle, Tabel 4.15 untuk container Uncontainer dan Karantina, Tabel 4.16 untuk

container uncontainer dan *Behandle*, Tabel 4.17 untuk *container Reefer* dan Karantina dan Tabel 4.18 untuk *container Reefer* dan *Behandle*.



Gambar 4.19 Hasil *gantt chart* dari data log PT TPS

Tabel 4.13 HT Truck 001 untuk *container Dry* dan Karantina

Container	*Start*	*Stop*	*Pr.Tm*
ID025	6	24	18
ID010	24	40	16
ID013	40	59	19
ID028	59	76	17
ID019	76	95	19
ID022	95	115	20
ID001	115	135	20
ID004	135	153	18
ID007	153	171	18
ID016	171	191	20
ID058	191	209	18
ID031	209	225	16
ID046	225	241	16
ID061	241	257	16
ID043	257	277	20
ID052	277	296	19
ID055	296	313	17
ID040	313	329	16
ID034	329	349	20
ID037	349	368	19
ID049	368	388	20
ID064	388	404	16
ID085	404	425	21
ID091	425	442	17
ID094	442	458	16
ID070	458	475	17
ID067	475	491	16
ID079	491	509	18
ID076	509	526	17

ID073	526	542	16
ID082	542	557	15
ID088	557	573	16
ID097	573	593	20
ID100	593	607	14
Total			601

Tabel 4.14 HT Truck 002 untuk *container Dry* dan *Behandle*

Container	***Start Dur***	***Stop Dur***	***Processing Time***
ID025	24	43	19
ID010	43	61	18
ID013	61	79	18
ID028	79	96	17
ID019	96	113	17
ID022	115	133	18
ID001	135	154	19
ID004	154	171	17
ID007	171	190	19
ID016	191	208	17
ID058	209	225	16
ID031	225	243	18
ID046	243	258	15
ID061	258	274	16
ID043	**277**	**295**	**18**
ID052	296	312	16
ID055	313	329	16
ID040	329	347	18
ID034	349	366	17
ID037	368	384	16
ID049	**388**	**407**	**19**
ID064	407	421	14
ID085	**425**	**439**	**14**
ID091	**442**	**459**	**17**
ID094	459	478	19
ID070	478	495	17
ID067	495	511	16
ID079	511	524	13
ID076	526	545	19
ID073	545	562	17
ID082	562	574	12
ID088	574	590	16
ID097	**593**	**613**	**20**
ID100	613	635	22

Total	580

Tabel 4.15 HT Truck 003 untuk *container Uncontainer* dan Karantina

Container	***Start***	***Stop***	***Pr.Tm***
ID020	5	22	17
ID014	22	40	18
ID026	40	56	16
ID008	56	74	18
ID011	74	92	18
ID023	92	111	19
ID029	111	125	14
ID032	125	139	14
ID002	139	157	18
ID005	157	173	16
ID017	173	189	16
ID053	189	207	18
ID041	207	224	17
ID059	224	242	18
ID044	242	261	19
ID062	261	277	16
ID065	277	292	15
ID035	292	310	18
ID047	310	324	14
ID038	324	340	16
ID056	340	358	18
ID050	358	374	16
ID074	374	394	20
ID086	394	409	15
ID077	409	430	21
ID092	430	447	17
ID068	447	461	14
ID071	461	481	20
ID095	481	498	17
ID080	498	516	18
ID098	516	537	21
ID089	537	553	16
ID083	553	574	21
Total			569

Tabel 4.16 HT Truck 004 untuk *container Uncontainer* dan *Behandle*

Container	**Start Dur**	**Stop Dur**	**Processing Time**
ID020	22	41	19
ID014	41	56	15

ID026	56	75	19
ID008	75	95	20
ID011	95	113	18
ID023	113	129	16
ID029	129	144	15
ID032	144	162	18
ID002	162	182	20
ID005	182	202	20
ID017	202	216	14
ID053	216	232	16
ID041	232	250	18
ID059	250	266	16
ID044	266	281	15
ID062	281	299	18
ID065	299	315	16
ID035	315	335	20
ID047	335	351	16
ID038	351	371	20
ID056	371	390	19
ID050	390	408	18
ID074	408	426	18
ID086	426	444	18
ID077	444	465	21
ID092	465	485	20
ID068	485	501	16
ID071	501	524	23
ID095	524	539	15
ID080	539	559	20
ID098	559	577	18
ID089	577	595	18
ID083	595	610	15
Total			588

Tabel 4.17 HT Truck 005 untuk *container Reefer* dan Karantina

Container	***Start***	***Stop***	***Pr.Tm***
ID003	6	25	19
ID009	25	42	17
ID015	42	60	18
ID033	60	81	21
ID012	81	100	19
ID018	100	118	18
ID021	118	134	16
ID030	134	152	18

ID027	152	172	20
ID024	172	192	20
ID036	192	216	24
ID006	216	236	20
ID066	236	255	19
ID042	255	272	17
ID048	272	289	17
ID051	289	307	18
ID063	307	322	15
ID054	322	340	18
ID060	340	355	15
ID057	355	375	20
ID069	375	394	19
ID081	394	412	18
ID096	412	428	16
ID039	428	445	17
ID099	445	462	17
ID084	462	482	20
ID090	482	499	17
ID045	499	517	18
ID075	517	532	15
ID087	532	547	15
ID093	547	564	17
ID072	564	584	20
ID078	584	604	20
Total			598

Tabel 4.18 HT Truck 006 untuk *container Reefer* dan *Behandle*

Container	*Start Dur*	*Stop Dur*	*Processing Time*
ID003	25	41	16
ID009	42	61	19
ID015	61	77	16
ID033	**81**	**100**	**19**
ID012	100	117	17
ID018	118	137	19
ID021	137	158	21
ID030	158	177	19
ID027	177	198	21
ID024	198	216	18
ID036	216	231	15
ID006	**236**	**253**	**17**
ID066	255	274	19
ID042	274	292	18

ID048	292	308	16
ID051	308	324	16
ID063	324	343	19
ID054	343	362	19
ID060	362	379	17
ID057	379	396	17
ID069	396	417	21
ID081	417	434	17
ID096	434	452	18
ID039	452	472	20
ID099	472	488	16
ID084	488	504	16
ID090	504	520	16
ID045	520	540	20
ID075	540	556	16
ID087	556	572	16
ID093	572	592	20
ID072	592	609	17
ID078	609	628	19
Total			590

Dari hasil metode FCFS, terlihat bahwa *Container Dry* dan *Behandle* serta *container Reefer* dan *Behandle* mengalami waktu *waiting time* yang lama sehingga pada saat demikian, RTGC berubah fungsi membantu mengantarkan *container* dari *yard* ke *Behandle*. Hasil *scheduling* RTGC dan HT Truck akan berubah, RTCG akan bertambah waktu durasi eksekusinya dan HT Truck akan berkurang sehingga dapat melanjutkan proses pengangkutan ke *container* selanjutnya. Tabel 4.19 menunjukkan hasil HT Truck 02 dan Departemen *Behandle* setelah *scheduling* selesai. Tabel 4.20 menunjukkan hasil pertambahan waktu RTGC setelah *scheduling* selesai dan rangkuman hasil *scheduling* untuk HT truck 02 dan departemen *Behandle* dijelaskan pada Tabel 4.21.

Tabel 4.19 Hasil HT Truck 02 dan departemen *Behandle* setelah *scheduling*

Container	Start Dur	Stop Dur	Processing Time
25	6	25	19
10	25	43	18
13	43	61	18
28	61	78	17
19	78	95	17
22	97	115	18
1	117	136	19
4	136	153	17
7	153	172	19
16	173	190	17

58	191	207	16
31	207	225	18
46	225	240	15
61	240	256	16
52	257	273	16
55	274	290	16
40	290	308	18
34	310	327	17
37	329	345	16
64	345	359	14
94	359	378	19
70	378	395	17
67	395	411	16
79	411	424	13
76	426	445	19
73	445	462	17
82	462	474	12
88	474	490	16
100	490	512	22
			492

Tabel 4.20 Hasil pertambahan waktu RTGC setelah *scheduling*

RTGC 10	*Start Dur*	*Stop Dur*	*Processing Time*
ID010	1	7	6
ID043	7	22	6
ID076	29	36	7
RTGC 16	*Start Dur*	*Stop Dur*	*Processing Time*
ID016	1	10	9
ID049	10	26	5
ID082	26	31	5
RTGC 19	*Start Dur*	*Stop Dur*	*Processing Time*
ID019	1	8	7
ID052	8	13	5
ID085	13	24	4
RTGC 25	*Start Dur*	*Stop Dur*	*Processing Time*
ID025	1	6	5
ID058	6	10	4
ID091	10	26	8
RTGC 31	*Start Dur*	*Stop Dur*	*Processing Time*
ID031	1	12	11
ID064	12	17	5
ID097	17	36	5

Tabel 4.21 Rangkuman hasil *scheduling* untuk HT truck 02 dan Departemen *Behandle*

		Total	*End time*
Quarantine	HTT 01	601	607
Behandle	HTT 02	492	512
	Awal	*Additional*	Akhir
RTGC 10	19	9	28
RTGC 16	19	11	30
RTGC 19	16	7	23
RTGC 25	17	8	25
RTGC 31	21	14	35

Langkah yang sama juga diterapkan pada HT Truck 06 dan Departemen *Behandle* untuk mengatasi waktu *waiting time* yang lama untuk setiap pengangkutan *container*. Tabel 4.22 menunjukkan hasil HT Truck 06 dan Departemen *Behandle* setelah *scheduling* selesai. Tabel 4.23 menunjukkan hasil pertambahan waktu RTGC setelah *scheduling* selesai dan rangkuman hasil *scheduling* untuk HT truck 06 dan Departemen *Behandle* dijelaskan pada Tabel 4.24.

Tabel 4.22 Hasil HT Truck 06 dan departemen *Behandle* setelah *scheduling*

Container	*Start Dur*	*Stop Dur*	*Processing Time*
ID003	6	41	16
ID009	42	61	19
ID015	61	77	16
ID012	77	94	17
ID018	94	113	19
ID021	114	135	21
ID030	135	154	19
ID027	154	175	21
ID024	175	193	18
ID036	193	208	15
ID066	210	229	19
ID042	229	247	18
ID048	247	263	16
ID051	263	279	16
ID063	279	298	19
ID054	298	317	19
ID060	317	334	17
ID057	334	351	17
ID069	351	372	21
ID081	372	389	17
ID096	389	407	18
ID039	407	427	20
ID099	427	443	16

ID084	443	459	16
ID090	459	475	16
ID045	475	495	20
ID075	495	511	16
ID087	511	527	16
ID093	527	547	20
ID072	547	564	17
ID078	564	583	19
Total			554

Tabel 4.23 Hasil pertambahan waktu RTGC setelah *scheduling*

RTGC 33	*Start Dur*	*Stop Dur*	*Processing Time Awal*	*FMS*
ID033	1	18	6	11
ID066	18	23	5	
ID099	23	30	7	
RTGC 06	*Start Dur*	*Stop Dur*	*Processing Time Awal*	*FMS*
ID006	1	20	11	8
ID039	20	27	7	
ID072	27	31	4	

Tabel 4.24 Hasil scheduling untuk HT truck 06 dan departemen behandle

		Total	*End time*
Quarantine	HTT 05	598	604
Behandle	HTT 06	554	583
	Awal	*Additional*	Akhir
RTGC 33	18	11	29
RTGC 06	22	8	30

Setelah dilakukan *scheduling* untuk hari pertama di bulan Februari, maka dapat dirangkum setiap hari selama 28 hari di bulan Februari berapa jumlah total *container* per tipe, jumlah waktu eksekusi per departemen, berapa kali RTGC berubah fungsi sehingga rata-ratanya diperoleh. Hasil *scheduling* dan rata-rata RTGC yang berubah fungsi untuk *container reefer* yaitu 2-3 kali dalam sebulan, untuk *container dry* sebanyak 8-9 kali per bulan dan untuk *container uncontainer* sebanyak 0-1 kali dalam sebulan.

Hari ke-	*REEFER*					*DRY*				*UNCONTAINER*			
	Departemen	HT Truck	*After Sche.*	*Bef sche.*	Jml Container	HT Truck	*After Sche.*	*Bef sche.*	Jml Container	HT Truck	*After Sche.*	*Bef sche.*	Jml Container
1	*Quarantine*	HTT 05	598	604	33	HTT 01	601	607	34	HTT 03	569	607	33
	Behandle	HTT 06	554	583		HTT 02	492	512		HTT 04	588	593	
	RTGC yang FMS	2				5				0			
2	*Quarantine*	HTT 05	534	540	30	HTT 01	997	1011	58	HTT 03	199	213	12
	Behandle	HTT 06	494	520		HTT 02	811	848		HTT 04	192	207	
	RTGC yang FMS	2				9				1			
3	*Quarantine*	HTT 05	569	593	31	HTT 01	897	900	51	HTT 03	297	311	18
	Behandle	HTT 06	519	568		HTT 02	733	757		HTT 04	287	303	
	RTGC yang FMS	3				6				1			
4	*Quarantine*	HTT 05	582	599	31	HTT 01	892	902	52	HTT 03	295	314	17
	Behandle	HTT 06	535	576		HTT 02	725	756		HTT 04	285	306	
	RTGC yang FMS	3				8				1			
5	*Quarantine*	HTT 05	464	487	26	HTT 01	1021	1044	60	HTT 03	231	246	14

	Behandle	HTT 06	417	468		HTT 02	828	876		HTT 04	223	240	
	RTGC yang FMS	3				12				1			
6	*Quarantine*	HTT 05	532	541	29	HTT 01	1005	1013	58	HTT 03	214	230	13
	Behandle	HTT 06	482	516		HTT 02	819	850		HTT 04	206	224	
	RTGC yang FMS	2				7				1			
7	*Quarantine*	HTT 05	643	654	36	HTT 01	867	894	50	HTT 03	258	274	14
	Behandle	HTT 06	593	634		HTT 02	707	754		HTT 04	250	268	
	RTGC yang FMS	3				11				1			
8	*Quarantine*	HTT 05	595	607	33	HTT 01	988	995	56	HTT 03	198	202	11
	Behandle	HTT 06	549	588		HTT 02	808	838		HTT 04	192	197	
	RTGC yang FMS	3				7				0			
9	*Quarantine*	HTT 05	591	603	33	HTT 01	1005	1012	57	HTT 03	171	182	10
	Behandle	HTT 06	543	580		HTT 02	822	852		HTT 04	165	178	
	RTGC yang FMS	3				8				1			
10	*Quarantine*	HTT 05	467	481	27	HTT 01	1027	1044	59	HTT 03	243	261	14

	Behandle	HTT 06	422	461		HTT 02	838	879		HTT 04	235	255	
	RTGC yang FMS	3				10				2			
11	*Quarantine*	HTT 05	384	393	22	HTT 01	1096	1109	63	HTT 03	265	282	15
	Behandle	HTT 06	336	370		HTT 02	894	933		HTT 04	256	276	
	RTGC yang FMS	2				9				1			
12	*Quarantine*	HTT 05	587	600	33	HTT 01	921	931	53	HTT 03	255	266	14
	Behandle	HTT 06	539	575		HTT 02	751	782		HTT 04	247	260	
	RTGC yang FMS	3				7				0			
13	*Quarantine*	HTT 05	586	597	32	HTT 01	993	1000	56	HTT 03	202	215	12
	Behandle	HTT 06	543	572		HTT 02	813	844		HTT 04	195	209	
	RTGC yang FMS	2				8				0			
14	*Quarantine*	HTT 05	572	592	31	HTT 01	990	1002	56	HTT 03	220	230	13
	Behandle	HTT 06	541	572		HTT 02	810	846		HTT 04	213	224	
	RTGC yang FMS	2				9				0			
15	*Quarantine*	HTT 05	525	538	29	HTT 01	1031	1037	59	HTT 03	218	237	12

	Behandle	HTT 06	482	513		HTT 02	841	872		HTT 04	211	232	
	RTGC yang FMS	2				7				2			
16	*Quarantine*	HTT 05	431	440	25	HTT 01	1088	1101	64	HTT 03	196	210	11
	Behandle	HTT 06	385	418		HTT 02	883	922		HTT 04	189	205	
	RTGC yang FMS	2				9				1			
17	*Quarantine*	HTT 05	387	394	22	HTT 01	1102	1113	63	HTT 03	261	278	15
	Behandle	HTT 06	343	371		HTT 02	900	936		HTT 04	252	272	
	RTGC yang FMS	2				8				1			
18	*Quarantine*	HTT 05	436	444	25	HTT 01	1024	1033	60	HTT 03	257	274	15
	Behandle	HTT 06	387	421		HTT 02	832	865		HTT 04	248	268	
	RTGC yang FMS	2				8				1			
19	*Quarantine*	HTT 05	645	658	38	HTT 01	893	905	51	HTT 03	194	207	11
	Behandle	HTT 06	596	637		HTT 02	730	763		HTT 04	188	202	
	RTGC yang FMS	3				9				0			
20	*Quarantine*	HTT 05	592	610	34	HTT 01	918	934	54	HTT 03	218	235	12

	Behandle	HTT 06	545	590		HTT 02	745	783		HTT 04	211	230	
	RTGC yang FMS	3				9				1			
21	*Quarantine*	HTT 05	326	336	19	HTT 01	1248	1252	70	HTT 03	191	198	11
	Behandle	HTT 06	276	314		HTT 02	1024	1056		HTT 04	185	193	
	RTGC yang FMS	3				7				0			
22	*Quarantine*	HTT 05	563	567	30	HTT 01	997	1008	57	HTT 03	232	247	13
	Behandle	HTT 06	517	548		HTT 02	814	849		HTT 04	225	241	
	RTGC yang FMS	2				14				0			
23	*Quarantine*	HTT 05	562	588	30	HTT 01	1028	1036	59	HTT 03	177	188	11
	Behandle	HTT 06	518	568		HTT 02	839	871		HTT 04	171	183	
	RTGC yang FMS	4				9				0			
24	*Quarantine*	HTT 05	380	395	21	HTT 01	1144	1148	67	HTT 03	206	220	12
	Behandle	HTT 06	337	374		HTT 02	929	961		HTT 04	199	215	
	RTGC yang FMS	3				8				1			
25	*Quarantine*	HTT 05	356	364	20	HTT 01	1224	1244	71	HTT 03	169	179	9

	Behandle	HTT 06	312	344		HTT 02	996	1046		HTT 04	164	175	
	RTGC yang FMS	2				12				0			
26	*Quarantine*	HTT 05	570	597	30	HTT 01	997	1008	58	HTT 03	217	231	12
	Behandle	HTT 06	520	574		HTT 02	811	846		HTT 04	210	225	
	RTGC yang FMS	4				8				1			
27	*Quarantine*	HTT 05	559	596	29	HTT 01	1021	1030	60	HTT 03	190	201	11
	Behandle	HTT 06	510	573		HTT 02	829	862		HTT 04	184	196	
	RTGC yang FMS	4				8				0			
28	*Quarantine*	HTT 05	327	336	18	HTT 01	1223	1238	71	HTT 03	198	210	11
	Behandle	HTT 06	279	317		HTT 02	995	1040		HTT 04	191	205	
	RTGC yang FMS	3				11				1			
Rata-rata FMS RTGC		2,678571429			Total Cont	8,678571429			Total Cont	0,678571429			Total
					797				1627				376

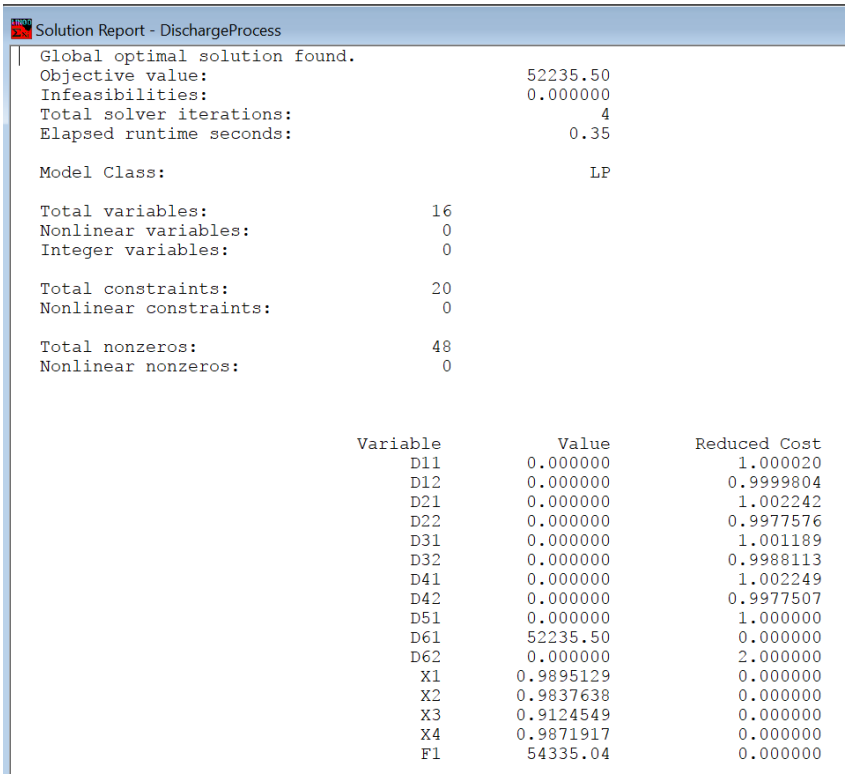
4.5 Optimasi dengan *Goal programming*

4.5.1 Model matematika untuk Blok Proses *Discharge*

Berdasarkan waktu dan biaya yang telah dijabarkan pada Subbab 3.7, yang digunakan untuk optimasi pada blok proses *discharge* adalah total dari waktu rata-rata durasi eksekusi setiap aktivitas sebagai *goal* dan yang akan diminimalkan adalah total waktu maksimum dari setiap aktivitas pada blok proses *discharge* yang meliputi *Vessel Berthing Process, Discharge container, Bring container to yard* dan *Stack container in yard*. Model matematika untuk optimasi dengan *goal programming* blok proses *discharge* dijabarkan pada Tabel 4.25. Model matematika tersebut diinput ke kakas bantu Lingo untuk membantu mengolah data sehingga memperoleh hasil sesuai *goal*. Gambar 4.20 menunjukkan hasil *output* dari Lingo.

Tabel 4.25 Model matematika untuk blok proses *discharge*

```
Min=(d11+d12+d21+d22+d31+d32+d41+d42)+(d51)+(d61+d62);
51130.4*x1+d11-d12=50594.19;
445.955*x2+d21-d22=438.7144;
841.229*x3+d31-d32=767.5835;
444.585*x4+d41-d42=438.8906;
54330.8*x1+502.813*x2+61.7078*x3+23.3576*x4+d51=f1;
x1+x2+x3+x4+d61-d62=52239.3766;
x1>=0;
d11>=0;
d12>=0;
x2>=0;
d21>=0;
d22>=0;
x3>=0;
d31>=0;
d32>=0;
x4>=0;
d41>=0;
d42>=0;
d51>=0;
end
```



Solution Report - DischargeProcess

```
Global optimal solution found.
Objective value:                              52235.50
Infeasibilities:                              0.000000
Total solver iterations:                             4
Elapsed runtime seconds:                          0.35

Model Class:                                        LP

Total variables:                    16
Nonlinear variables:                 0
Integer variables:                   0

Total constraints:                  20
Nonlinear constraints:               0

Total nonzeros:                     48
Nonlinear nonzeros:                  0


                         Variable           Value        Reduced Cost
                              D11        0.000000            1.000020
                              D12        0.000000           0.9999804
                              D21        0.000000            1.002242
                              D22        0.000000           0.9977576
                              D31        0.000000            1.001189
                              D32        0.000000           0.9988113
                              D41        0.000000            1.002249
                              D42        0.000000           0.9977507
                              D51        0.000000            1.000000
                              D61        52235.50            0.000000
                              D62        0.000000            2.000000
                               X1       0.9895129            0.000000
                               X2       0.9837638            0.000000
                               X3       0.9124549            0.000000
                               X4       0.9871917            0.000000
                               F1        54335.04            0.000000
```

Gambar 4.20 Hasil *output* Lingo untuk blok proses *discharge*

Berdasarkan *output* dari Lingo, *goal* berhasil dicapai dengan penyimpangan positif kearah *goal* sebesar 52235,50 dan biaya yang akan dikeluarkan oleh blok proses *discharge* jika waktu berhasil diminimumkan adalah USD 54335.04.

4.5.2 Model matematika untuk Blok Proses Karantina

Berdasarkan waktu dan biaya yang telah dijabarkan pada Subbab 3.7, yang digunakan untuk optimasi pada blok proses karantina adalah total dari waktu rata-rata durasi eksekusi setiap aktivitas sebagai *goal* dan yang akan diminimalkan adalah total waktu maksimum dari setiap aktivitas pada blok proses karantina yang dibagi menjadi kategori barang dan dokumen. Pemisahan dalam mengoptimasi adalah karena setelah dianalisis dengan menggunakan kakas bantu Disco, dokumen lebih lama prosesnya dibandingkan dengan proses barang. Aktivitas pada kategori dokumen di blok proses karantina yang akan di optimasi meliputi *Verification Documen Quarantine, Create Job Order Document Quarantine, Create Document KH/KT and Send Certificate KH/KT Info*. Sedangkan aktivitas untuk kategori barang yang akan di optimasi adalah *Bring container from yard to quarantine, stack container in quarantine area, check goods quarantine* dan *stack container in yard from quarantine*.

Model matematika untuk optimasi dengan *goal programming* blok proses karantina dijabarkan pada Tabel 4.26 untuk kategori dokumen dan Tabel 4.27 untuk kategori barang. Model matematika tersebut diinput ke kakas bantu Lingo untuk

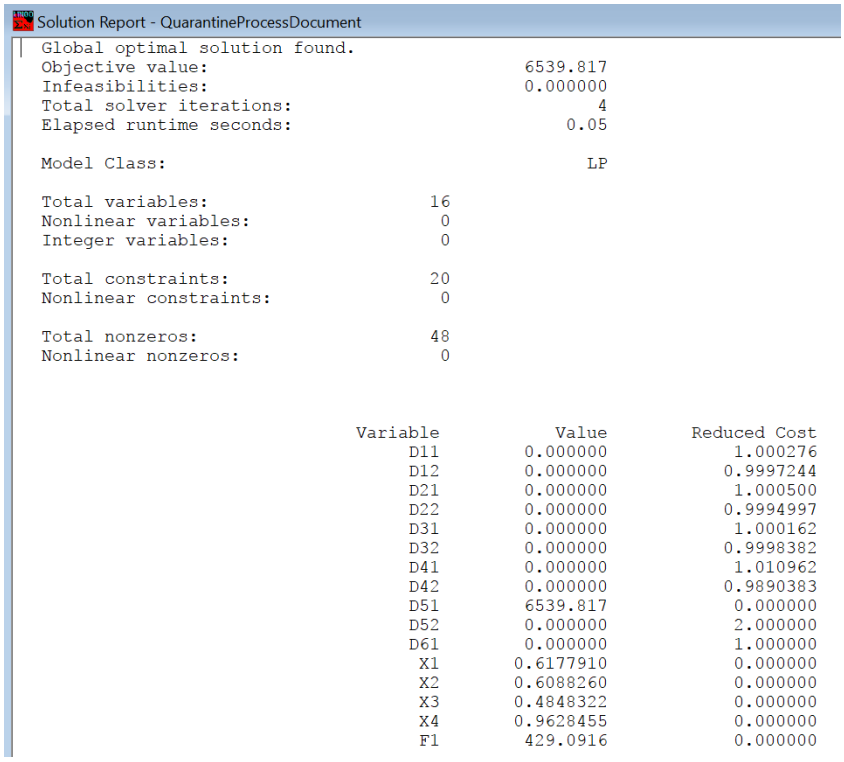
membantu mengolah data sehingga memperoleh hasil sesuai *goal*. Gambar 4.21 dan Gambar 4.22 menunjukkan hasil *output* dari Lingo kategori dokumen dan barang.

Tabel 4.26 Model matematika untuk blok proses karantina kategori dokumen

```
Min=(d11+d12+d21+d22+d31+d32+d41+d42)+(d51+d52)+(d61);
3629.101*x1+d11-d12=2242.026;
1998.676*x2+d21-d22=1216.846;
6179.008*x3+d31-d32=2995.782;
91.22642*x4+d41-d42=87.83695;
x1+x2+x3+x4+d51-d52=6542.491;
177.3335*x1+177.1322*x2+413.1588*x3+11.82036*x4+d61=f1;
x1>=0;
d11>=0;
d12>=0;
x2>=0;
d21>=0;
d22>=0;
x3>=0;
d31>=0;
d32>=0;
x4>=0;
d41>=0;
d42>=0;
d51>=0;
end
```

Tabel 4.27 Model matematika untuk blok proses karantina kategori barang

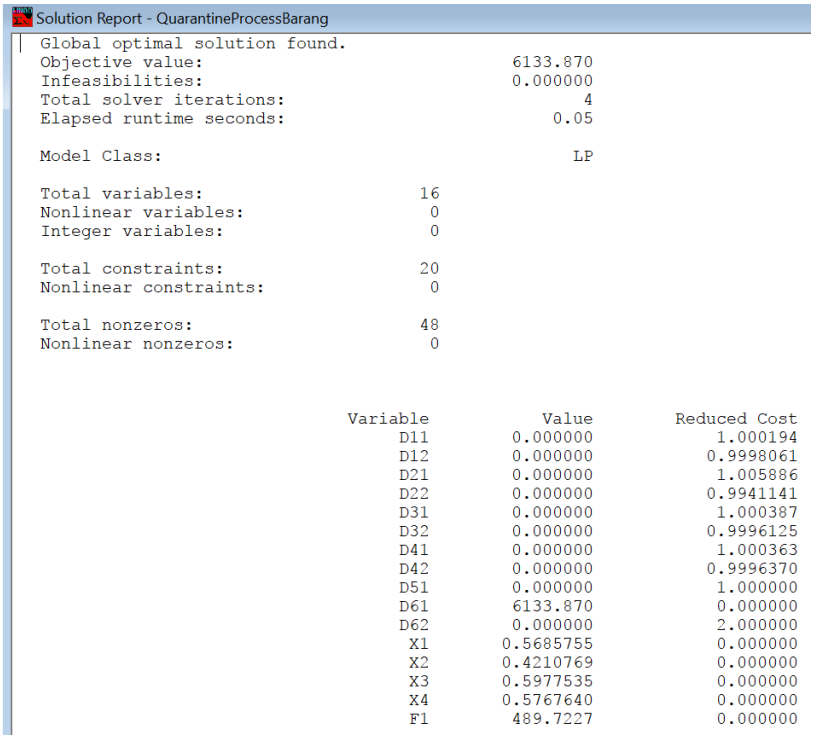
```
Min=(d11+d12+d21+d22+d31+d32+d41+d42)+(d51)+(d61+d62);
5158.487*x1+d11-d12=2932.989226;
169.8976*x2+d21-d22=71.53996;
2580.858*x3+d31-d32=1542.716835;
2754.66*x4+d41-d42=1588.78861;
177.1698*x1+177.0851*x2+354.8908*x3+177.3423*x4+d51=f1;
x1+x2+x3+x4+d61-d62=6136.0346;
x1>=0;
d11>=0;
d12>=0;
x2>=0;
d21>=0;
d22>=0;
x3>=0;
d31>=0;
d32>=0;
x4>=0;
d41>=0;
d42>=0;
d51>=0;
end
```



Solution Report - QuarantineProcessDocument

```
Global optimal solution found.
Objective value:                              6539.817
Infeasibilities:                              0.000000
Total solver iterations:                             4
Elapsed runtime seconds:                          0.05

Model Class:                                        LP

Total variables:                    16
Nonlinear variables:                 0
Integer variables:                   0

Total constraints:                  20
Nonlinear constraints:               0

Total nonzeros:                     48
Nonlinear nonzeros:                  0


                     Variable           Value        Reduced Cost
                          D11        0.000000            1.000276
                          D12        0.000000            0.9997244
                          D21        0.000000            1.000500
                          D22        0.000000            0.9994997
                          D31        0.000000            1.000162
                          D32        0.000000            0.9998382
                          D41        0.000000            1.010962
                          D42        0.000000            0.9890383
                          D51        6539.817            0.000000
                          D52        0.000000            2.000000
                          D61        0.000000            1.000000
                           X1        0.6177910           0.000000
                           X2        0.6088260           0.000000
                           X3        0.4848322           0.000000
                           X4        0.9628455           0.000000
                           F1        429.0916            0.000000
```

Gambar 4.21 Hasil *output* Lingo untuk blok proses karantina kategori dokumen



Solution Report - QuarantineProcessBarang

```
Global optimal solution found.
Objective value:                              6133.870
Infeasibilities:                              0.000000
Total solver iterations:                             4
Elapsed runtime seconds:                          0.05

Model Class:                                        LP

Total variables:                    16
Nonlinear variables:                 0
Integer variables:                   0

Total constraints:                  20
Nonlinear constraints:               0

Total nonzeros:                     48
Nonlinear nonzeros:                  0


                     Variable           Value        Reduced Cost
                          D11        0.000000            1.000194
                          D12        0.000000            0.9998061
                          D21        0.000000            1.005886
                          D22        0.000000            0.9941141
                          D31        0.000000            1.000387
                          D32        0.000000            0.9996125
                          D41        0.000000            1.000363
                          D42        0.000000            0.9996370
                          D51        0.000000            1.000000
                          D61        6133.870            0.000000
                          D62        0.000000            2.000000
                           X1        0.5685755           0.000000
                           X2        0.4210769           0.000000
                           X3        0.5977535           0.000000
                           X4        0.5767640           0.000000
                           F1        489.7227            0.000000
```

Gambar 4.22 Hasil *output* Lingo untuk blok proses karantina kategori barang

Berdasarkan *output* dari Lingo, *goal* berhasil dicapai dengan penyimpangan positif kearah *goal* sebesar 6539,817 dan biaya yang akan dikeluarkan oleh blok proses karantina jika waktu berhasil diminimumkan adalah USD 429,0916 untuk kategori dokumen. Sedangkan, *goal* berhasil dicapai dengan penyimpangan positif kearah *goal* sebesar 6133,87 dan biaya yang akan dikeluarkan oleh blok proses karantina jika waktu berhasil diminimumkan adalah USD 489,7227 untuk kategori barang.

4.5.3 Model matematika untuk Blok Proses *Behandle*

Berdasarkan waktu dan biaya yang telah dijabarkan pada Subbab 3.7, yang digunakan untuk optimasi pada blok proses *behandle* adalah total dari waktu rata-rata durasi eksekusi setiap aktivitas sebagai *goal* dan yang akan diminimalkan adalah total waktu maksimum dari setiap aktivitas pada blok proses karantina yang dibagi menjadi kategori barang dan dokumen. Pemisahan dalam mengoptimasi adalah karena setelah dianalisis dengan menggunakan kakas bantu Disco, dokumen lebih lama prosesnya dibandingkan dengan proses barang. Aktivitas pada kategori dokumen di blok proses *behandle* yang akan di optimasi meliputi *Verification Documen Behandle, Create Job Order Document Behandle, Create Document LHP and Create Document SPPB.* Sedangkan aktivitas untuk kategori barang yang akan di optimasi adalah *Bring container from yard to behandle, stack container in behandle area, check goods behandle* dan *stack container in yard from behandle*.

Model matematika untuk optimasi dengan *goal programming* blok proses *behandle* dijabarkan pada Tabel 4.28 untuk kategori dokumen, Tabel 4.29 untuk kategori barang dan Tabel 4.30 untuk behandle jalur hijau. Model matematika tersebut diinput ke kakas bantu Lingo untuk membantu mengolah data sehingga memperoleh hasil sesuai *goal*. Gambar 4.23, Gambar 4.24, dan Gambar 4.25 menunjukkan hasil *output* dari Lingo kategori dokumen, barang dan *behandle* jalur hijau.

Tabel 4.28 Model matematika untuk blok proses behandle kategori dokumen

```
Min=(d11+d12+d21+d22+d31+d32+d41+d42)+(d51)+(d61+d62);
249.8915*x1+d11-d12=245.639;
182555.261*x2+d21-d22=107667.7328;
6283.420*x3+d31-d32=3520.588;
39855.234*x4+d41-d42=23193.208;
6.777*x1+465.535*x2+22445.197*x3+2656.990*x4+d51=f1;
x1+x2+x3+x4+d61-d62=134627.1691;
x1>=0;
d11>=0;
d12>=0;
x2>=0;
d21>=0;
d22>=0;
x3>=0;
d31>=0;
d32>=0;
x4>=0;
d41>=0;
```

```
d42>=0;
d51>=0;
end
```

Tabel 4.29 Model matematika untuk blok proses behandle kategori barang

```
Min=(d11+d12+d21+d22+d31+d32+d41+d42)+(d51)+(d61+d62);
13816.8046*x1+d11-d12=8156.976;
15681.0682*x2+d21-d22=8260.8539;
3610.4659*x3+d31-d32=1868.7563;
187.8387*x4+d41-d42=83.7543;
465.7555*x1+468.8945*x2+1104.6365*x3+459.6799*x4+d51=f1;
x1+x2+x3+x4+d61-d62=18370.3406;
x1>=0;
d11>=0;
d12>=0;
x2>=0;
d21>=0;
d22>=0;
x3>=0;
d31>=0;
d32>=0;
x4>=0;
d41>=0;
d42>=0;
d51>=0;
end
```

Tabel 4.30 Model matematika untuk blok proses behandle jalur hijau

```
Min=(d11+d12+d21+d22)+(d31)+(d41+d42);
249.8915*x1+d11-d12=245.639;
39855.2084*x2+d21-d22=23193.20843;
6.777*x1+2658.3441*x2+d31=f1;
x1+x2+d41-d42=23438.84779;
x1>=0;
d11>=0;
d12>=0;
x2>=0;
d21>=0;
d22>=0;
end
```

Solution Report - BehandleProcessDocument

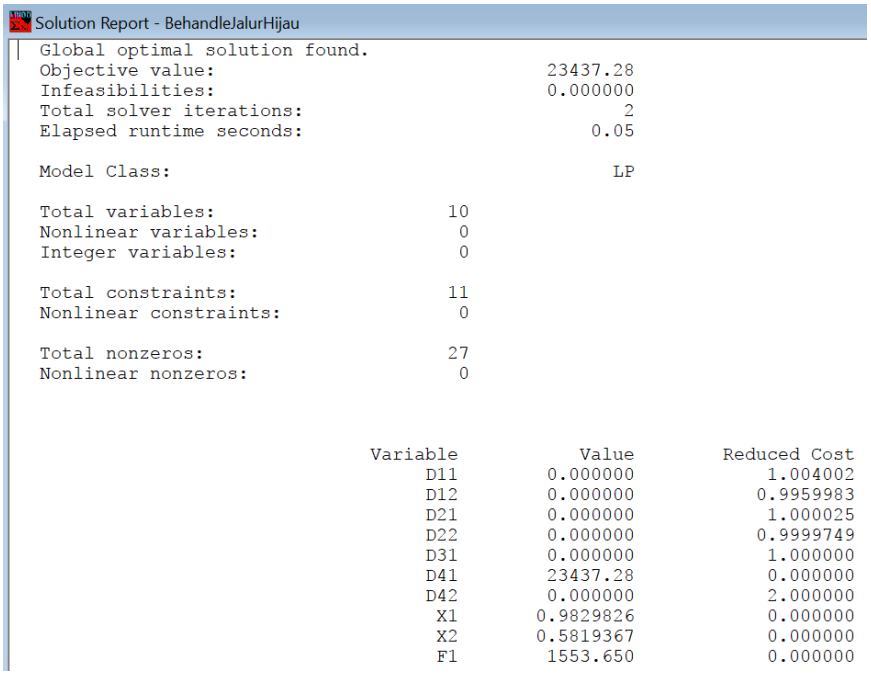
```
Global optimal solution found.
Objective value:                              134624.5
Infeasibilities:                              0.000000
Total solver iterations:                             4
Elapsed runtime seconds:                          0.05

Model Class:                                        LP

Total variables:                    16
Nonlinear variables:                 0
Integer variables:                   0

Total constraints:                  20
Nonlinear constraints:               0

Total nonzeros:                     48
Nonlinear nonzeros:                  0


                            Variable           Value        Reduced Cost
                                 D11        0.000000            1.004002
                                 D12        0.000000           0.9959983
                                 D21        0.000000            1.000005
                                 D22        0.000000           0.9999945
                                 D31        0.000000            1.000159
                                 D32        0.000000           0.9998409
                                 D41        0.000000            1.000025
                                 D42        0.000000           0.9999749
                                 D51        0.000000            1.000000
                                 D61        134624.5            0.000000
                                 D62        0.000000            2.000000
                                  X1       0.9829826            0.000000
                                  X2       0.5897816            0.000000
                                  X3       0.5602981            0.000000
                                  X4       0.5819363            0.000000
                                  F1        14403.42            0.000000
```

Gambar 4.23 Hasil *output* Lingo untuk blok proses behandle kategori dokumen

Solution Report - BehandleProcessBarang

```
Global optimal solution found.
Objective value:                              18368.26
Infeasibilities:                              0.000000
Total solver iterations:                             4
Elapsed runtime seconds:                          0.05

Model Class:                                        LP

Total variables:                    16
Nonlinear variables:                 0
Integer variables:                   0

Total constraints:                  20
Nonlinear constraints:               0

Total nonzeros:                     48
Nonlinear nonzeros:                  0


                            Variable           Value        Reduced Cost
                                 D11        0.000000            1.000072
                                 D12        0.000000           0.9999276
                                 D21        0.000000            1.000064
                                 D22        0.000000           0.9999362
                                 D31        0.000000            1.000277
                                 D32        0.000000           0.9997230
                                 D41        0.000000            1.005324
                                 D42        0.000000           0.9946763
                                 D51        0.000000            1.000000
                                 D61        18368.26            0.000000
                                 D62        0.000000            2.000000
                                  X1       0.5903663            0.000000
                                  X2       0.5268043            0.000000
                                  X3       0.5175942            0.000000
                                  X4       0.4458842            0.000000
                                  F1         1298.699           0.000000
```

Gambar 4.24 Hasil *output* Lingo untuk blok proses behandle kategori barang



Solution Report - BehandleJalurHijau

```
Global optimal solution found.
Objective value:                              23437.28
Infeasibilities:                              0.000000
Total solver iterations:                             2
Elapsed runtime seconds:                          0.05

Model Class:                                        LP

Total variables:                     10
Nonlinear variables:                  0
Integer variables:                    0

Total constraints:                   11
Nonlinear constraints:                0

Total nonzeros:                      27
Nonlinear nonzeros:                   0


                         Variable           Value        Reduced Cost
                              D11        0.000000            1.004002
                              D12        0.000000            0.9959983
                              D21        0.000000            1.000025
                              D22        0.000000            0.9999749
                              D31        0.000000            1.000000
                              D41        23437.28            0.000000
                              D42        0.000000            2.000000
                               X1        0.9829826           0.000000
                               X2        0.5819367           0.000000
                               F1        1553.650            0.000000
```

Gambar 4.25 Hasil *output* Lingo untuk blok proses behandle jalur hijau

Berdasarkan *output* dari Lingo, *goal* berhasil dicapai dengan penyimpangan positif kearah *goal* sebesar 134624,5 dan biaya yang akan dikeluarkan oleh blok proses *behandle* jika waktu berhasil diminimumkan adalah USD 14403,42 untuk kategori dokumen. Sedangkan, *goal* berhasil dicapai dengan penyimpangan positif kearah *goal* sebesar 18368,26 dan biaya yang akan dikeluarkan oleh blok proses *behandle* jika waktu berhasil diminimumkan adalah USD 1298,699 untuk kategori barang. Dan untuk *behandle* jalur hijau *goal* juga berhasil dicapai dengan penyimpangan positif kearah *goal* sebesar 23437,28 dan biaya yang akan dikeluarkan oleh blok proses *behandle* jika waktu berhasil diminimumkan adalah USD 1553,65.

4.5.4 Model matematika untuk Blok Proses *Delivery*

Berdasarkan waktu dan biaya yang telah dijabarkan pada Subbab 3.7, yang digunakan untuk optimasi pada blok proses *delivery* adalah total dari waktu rata-rata durasi eksekusi setiap aktivitas sebagai *goal* dan yang akan diminimalkan adalah total waktu maksimum dari setiap aktivitas pada blok proses *delivery* yang meliputi *create job order document delivery, truck in, dispatch WQ delivery to CHE, Determine container type, determing dry, determining uncontainer, determining reefer, decide task before lift container, prepare tools, unplug reefer cable, lift on container truck, truck go to gate out, check container before truck out* dan *truck out*. Optimasi blok proses *delivery* dibedakan menjadi 3 bagian yaitu *container dry, container uncontainer*, dan *container dry.*

Model matematika untuk optimasi dengan *goal programming* blok proses *delivery* dijabarkan pada Tabel 4.31 untuk container dry, Tabel 4.32 untuk *container uncontainer*, dan Tabel 4.33 untuk *container reefer*. Model matematika tersebut *diinput* ke kakas bantu

Lingo untuk membantu mengolah data sehingga memperoleh hasil sesuai *goal*. Gambar 4.26, Gambar 4.27, dan Gambar 4.28 menunjukkan hasil output dari Lingo berturut-turut untuk *container dry, container uncontainer* dan *container reefer.*

Tabel 4.31 Model matematika untuk blok proses *delivery container dry*

```
Min=(d11+d12+d21+d22+d31+d32+d41+d42+d51+d52+d61+d62+d71+d72+d81+d82
+d91+d92+d101+d102)+(d111+d112)+(d121);
33958.30728*x1+d11-d12=24435.42372;
284646.3824*x2+d21-d22=68025.93198;
74.75233*x3+d31-d32=60.42906;
330.83046*x4+d41-d42=137.7133029;
176.58431*x5+d51-d52=106.6358361;
143.98476*x6+d61-d62=71.499935;
195.759245*x7+d71-d72=117.246195;
5469.806641*x8+d81-d82=2363.9795;
76.248073*x9+d91-d92=67.60261;
81.21234*x10+d101-d102=72.3447;
x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+d111-d112=95458.80691;
40.59155*x1+13.52148*x2+1.2042819*x3+2.7036198*x4+18.043195*x5+2.707
5818*x6+19.490719*x7+2.704978*x8+2.704948*x9+13.54043*x10+d121=f1;
x1>=0;
x2>=0;
x3>=0;
x4>=0;
x5>=0;
x6>=0;
x7>=0;
x8>=0;
x9>=0;
x10>=0;
d11>=0;
d12>=0;
d21>=0;
d22>=0;
d31>=0;
d32>=0;
d41>=0;
d42>=0;
d51>=0;
d52>=0;
d61>=0;
d62>=0;
d71>=0;
d72>=0;
d81>=0;
d82>=0;
d91>=0;
d92>=0;
d101>=0;
d102>=0;
d121=0;
end
```

Tabel 4.32 Model matematika untuk blok proses *delivery container uncontainer*

```
Min=(d11+d12+d21+d22+d31+d32+d41+d42+d51+d52+d61+d62+d71+d72+d81+d82
+d91+d92+d101+d102+d111+d112)+(d121+d122)+(d131);
33958.30728*x1+d11-d12=24435.42372;
284646.3824*x2+d21-d22=68025.93198;
74.75233*x3+d31-d32=60.42906;
330.83046*x4+d41-d42=137.7133029;
180*x5+d51-d52=98.215025;
143.98476*x6+d61-d62=71.499935;
2003.6667*x7+d71-d72=592.90472;
195.759245*x8+d81-d82=117.246195;
5469.806641*x9+d91-d92=2363.9795;
76.248073*x10+d101-d102=67.60261;
81.21234*x11+d111-d112=72.3447;
x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+x11+d121-d122=96043.29083;
40.59155*x1+13.52148*x2+1.2042819*x3+2.7036198*x4+18.30061*x5+2.7075
818*x6+54.28952*x7+19.490719*x8+2.704978*x9+2.704948*x10+13.54043*x1
1+d131=f1;
x1>=0;
x2>=0;
x3>=0;
x4>=0;
x5>=0;
x6>=0;
x7>=0;
x8>=0;
x9>=0;
x10>=0;
d11>=0;
d12>=0;
d21>=0;
d22>=0;
d31>=0;
d32>=0;
d41>=0;
d42>=0;
d51>=0;
d52>=0;
d61>=0;
d62>=0;
d71>=0;
d72>=0;
d81>=0;
d82>=0;
d91>=0;
d92>=0;
d101>=0;
d102>=0;
d111>=0;
d112>=0;
d131>=0;
end
```

Tabel 4.33 Model matematika untuk blok proses *delivery container reefer*

```
Min=(d11+d12+d21+d22+d31+d32+d41+d42+d51+d52+d61+d62+d71+d72+d81+d82
+d91+d92+d101+d102+d111+d112)+(d121+d122)+(d131);
33958.30728*x1+d11-d12=24435.42372;
```

```
284646.3824*x2+d21-d22=68025.93198;
74.75233*x3+d31-d32=60.42906;
330.83046*x4+d41-d42=137.7133029;
277.01826*x5+d51-d52=112.1029697;
143.98476*x6+d61-d62=71.499935;
381.640625*x7+d71-d72=312.724098;
195.759245*x8+d81-d82=117.246195;
5469.806641*x9+d91-d92=2363.9795;
76.248073*x10+d101-d102=67.60261;
81.21234*x11+d111-d112=72.3447;
x1+x2+x3+x4+x5+x6+x7+x8+x9+x10+x11+d121-d122=95776.99815;
40.59155*x1+13.52148*x2+1.2042819*x3+2.7036198*x4+22.85324*x5+2.7075
818*x6+14.91857*x7+19.490719*x8+2.704978*x9+2.704948*x10+13.54043*x1
1+d131=f1;
x1>=0;
x2>=0;
x3>=0;
x4>=0;
x5>=0;
x6>=0;
x7>=0;
x8>=0;
x9>=0;
x10>=0;
d11>=0;
d12>=0;
d21>=0;
d22>=0;
d31>=0;
d32>=0;
d41>=0;
d42>=0;
d51>=0;
d52>=0;
d61>=0;
d62>=0;
d71>=0;
d72>=0;
d81>=0;
d82>=0;
d91>=0;
d92>=0;
d101>=0;
d102>=0;
d111>=0;
d112>=0;
d131>=0;
end
```

Solution Report - DeliveryProcessDry

Variable	Value	Reduced Cost
D11	0.000000	1.000029
D12	0.000000	0.9999706
D21	0.000000	1.000004
D22	0.000000	0.9999965
D31	0.000000	1.013378
D32	0.000000	0.9866225
D41	0.000000	1.003023
D42	0.000000	0.9969773
D51	0.000000	1.005663
D52	0.000000	0.9943370
D61	0.000000	1.006945
D62	0.000000	0.9930548
D71	0.000000	1.005108
D72	0.000000	0.9948917
D81	0.000000	1.000183
D82	0.000000	0.9998172
D91	0.000000	1.013115
D92	0.000000	0.9868849
D101	0.000000	1.012313
D102	0.000000	0.9876866
D111	95452.71	0.000000
D112	0.000000	2.000000
D121	0.000000	0.000000
X1	0.7195713	0.000000
X2	0.2389840	0.000000
X3	0.8083903	0.000000
X4	0.4162655	0.000000
X5	0.6038806	0.000000
X6	0.4965799	0.000000
X7	0.5989306	0.000000
X8	0.4321870	0.000000
X9	0.8866140	0.000000
X10	0.8908092	0.000000
F1	74.08218	0.000000

Gambar 4.26 Hasil *output* Lingo untuk blok proses *delivery container dry*

Solution Report - DeliveryProcessUncontainer

Variable	Value	Reduced Cost
D11	0.000000	1.000029
D12	0.000000	0.9999706
D21	0.000000	1.000004
D22	0.000000	0.9999965
D31	0.000000	1.013378
D32	0.000000	0.9866225
D41	0.000000	1.003023
D42	0.000000	0.9969773
D51	0.000000	1.005556
D52	0.000000	0.9944444
D61	0.000000	1.006945
D62	0.000000	0.9930548
D71	0.000000	1.000499
D72	0.000000	0.9995009
D81	0.000000	1.005108
D82	0.000000	0.9948917
D91	0.000000	1.000183
D92	0.000000	0.9998172
D101	0.000000	1.013115
D102	0.000000	0.9868849
D111	0.000000	1.012313
D112	0.000000	0.9876866
D121	96036.96	0.000000
D122	0.000000	2.000000
D131	0.000000	1.000000
X1	0.7195713	0.000000
X2	0.2389840	0.000000
X3	0.8083903	0.000000
X4	0.4162655	0.000000
X5	0.5456390	0.000000
X6	0.4965799	0.000000
X7	0.2959099	0.000000
X8	0.5989306	0.000000
X9	0.4321870	0.000000
X10	0.8866140	0.000000
X11	0.8908092	0.000000
F1	89.23658	0.000000

Gambar 4.27 Hasil *output* Lingo untuk blok proses *delivery container uncontainer*

Solution Report - DeliveryProcessReefer

Variable	Value	Reduced Cost
D11	0.000000	1.000029
D12	0.000000	0.9999706
D21	0.000000	1.000004
D22	0.000000	0.9999965
D31	0.000000	1.013378
D32	0.000000	0.9866225
D41	0.000000	1.003023
D42	0.000000	0.9969773
D51	0.000000	1.003610
D52	0.000000	0.9963901
D61	0.000000	1.006945
D62	0.000000	0.9930548
D71	0.000000	1.002620
D72	0.000000	0.9973797
D81	0.000000	1.005108
D82	0.000000	0.9948917
D91	0.000000	1.000183
D92	0.000000	0.9998172
D101	0.000000	1.013115
D102	0.000000	0.9868849
D111	0.000000	1.012313
D112	0.000000	0.9876866
D121	95770.29	0.000000
D122	0.000000	2.000000
D131	0.000000	1.000000
X1	0.7195713	0.000000
X2	0.2389840	0.000000
X3	0.8083903	0.000000
X4	0.4162655	0.000000
X5	0.4046772	0.000000
X6	0.4965799	0.000000
X7	0.8194204	0.000000
X8	0.5989306	0.000000
X9	0.4321870	0.000000
X10	0.8866140	0.000000
X11	0.8908092	0.000000
F1	84.65901	0.000000

Gambar 4.28 Hasil *output* Lingo untuk blok proses *delivery container reefer*

Berdasarkan *output* dari Lingo, *goal* berhasil dicapai dengan penyimpangan positif kearah goal sebesar 95452,71 dan biaya yang akan dikeluarkan oleh blok proses *delivery* jika waktu berhasil diminimumkan adalah USD 74,08218 untuk *container dry*. Sedangkan, *goal* berhasil dicapai dengan penyimpangan positif kearah *goal* sebesar 96036,96 dan biaya yang akan dikeluarkan jika waktu berhasil diminimumkan adalah USD 89,23658 untuk *container uncontainer*. Dan terakhir untuk *container reefer goal* juga berhasil dicapai dengan penyimpangan positif kearah *goal* sebesar 95770,29 dan biaya yang akan dikeluarkan jika waktu berhasil diminimumkan adalah USD 74,08218.

4.5.5 Hasil Optimasi Waktu dan Biaya

Pada subbab ini akan dirangkum hasil optimasi waktu dan biaya dengan menggunakan *Goal programming* dan kakas bantu Lingo seperti pada Tabel 4.34.

HASIL OPTIMASI	*Time*						*Cost*			
	Goal	**Awal**	**Simpangan**	**Hasil Optimasi**	**Durasi yang bisa dioptimasi**	**Durasi Baru**	**Awal**	**Hasil Optimasi**	***Cost* hasil optimasi**	***Cost* Baru**
DISCHARGE (BARANG)	52239,37666	52862,16072	52235,5	Goal tercapai	622,7840563	52239,37666	57433,07027	54335,04	3098,030265	54335,04
QUARANTINE (DOKUMEN)	6542,49067	11898,01192	6539,817	Goal tercapai	5355,521248	6542,49067	829,7132027	429,0916	400,6216027	429,0916
QUARANTINE (BARANG)	6136,034632	12663,90324	6133,87	Goal tercapai	6527,868607	6136,034632	946,8966892	489,7227	457,1739892	489,7227
BEHANDLE (JALUR MERAH-DOKUMEN)	134627,1691	228943,8075	134624,5	Goal tercapai	94316,63847	134627,1691	29190,46865	14403,42	14787,04865	14403,42
BEHANDLE (JALUR MERAH-BARANG)	18370,3406	33296,1774	18368,26	Goal tercapai	14925,8368	18370,3406	2807,212871	1298,699	1508,513871	1298,699
BEHANDLE (JALUR HIJAU-DOKUMEN)	23438,84779	40105,12571	23437,28	Goal tercapai	16666,27792	23438,84779	2686,286384	1553,65	1132,636384	1553,65
DELIVERY (DRY)	95458,80691	325153,8679	95452,71	Goal tercapai	229695,061	95458,80691	129,517218	74,08218	55,43503796	74,08218
DELIVERY (UNCONTAINER)	96043,29083	327160,9502	96036,96	Goal tercapai	231117,6594	96043,29083	226,2523439	89,23658	137,0157639	89,23658
DELIVERY (REEFER)	95776,99815	325635,9425	95770,29	Goal tercapai	229858,9443	95776,99815	151,8007204	84,65901	67,14171038	84,65901

Tabel 4.34 Hasil optimasi waktu dan biaya dengan *Goal programming*

4.6 Evaluasi

Evaluasi ini dilakukan untuk proses *discovery* dengan algoritma modifikasi Alpha dengan membandingkan kinerja sistem yang dikembangkan dengan sistem lain. Perbandingan dilakukan dengan menggunakan hasil yang didapatkan pada hasil uji studi kasus.

4.6.1 Perbandingan Hasil *Fitness*

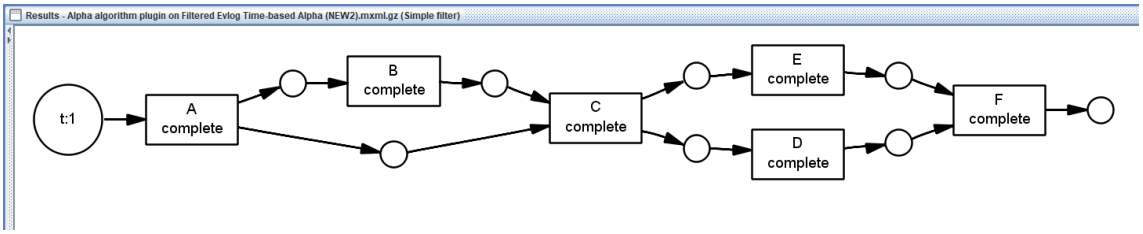
Dengan menggunakan formula perhitungan *fitness* pada Subbab 3.8 untuk memperoleh nilai *fitness*, lalu nilai fitness dari algoritma modifikasi Alpha yaitu *Modified Time-based Alpha Miner* dan Algoritma *Alpha* dibandingkan. Hasil perbandingan nilai *fitness* ditampilkan pada Tabel 4.35.

Tabel 4.35 Perbandingan hasil *fitness*

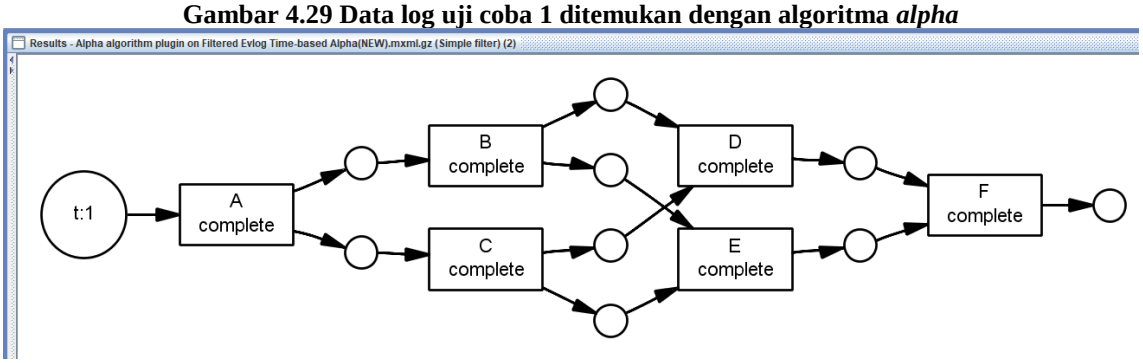
Data uji	Algoritma	Nilai *fitness*
Data log uji coba 1	*Modified Time-based Alpha Miner*	1,0
	Alpha	0,954
Data log uji coba 2	*Modified Time-based Alpha Miner*	1,0
	Alpha	0.977
Data log PT TPS dengan Anomali	*Modified Time-based Alpha Miner*	0,942
	Alpha	0,928

4.6.2 Perbandingan Model Proses, *Gateway* Paralel, dan Jumlah *Trace*

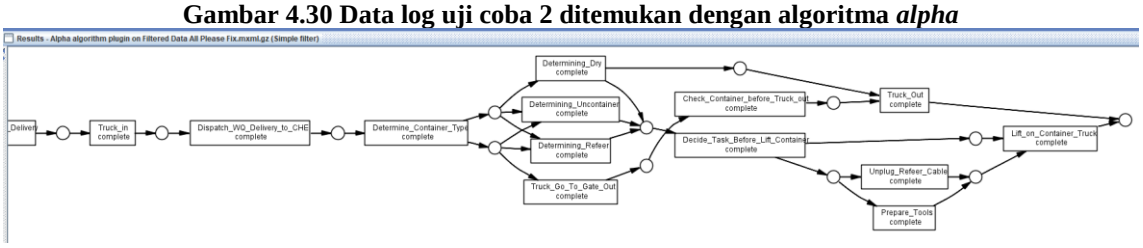
Model proses yang terbentuk dengan algoritma *Modified Time-based Alpha Miner* dengan Data log Uji Coba 1, Data log Uji Coba 2 dan Data log PT TPS seperti pada Gambar 4.5, Gambar 4.6 dan Gambar 4.7 pada Subbab 4.1.2. Dengan menggunakan data log yang sama, model proses akan di *discover* dengan algoritma *Alpha* dengan kakas bantu ProM seperti ditunjukkan pada Gambar 4.29 untuk data uji coba 1, Gambar 4.30 untuk data uji coba 2 dan Gambar 4.31 untuk data log PT TPS.



Gambar 4.29 Data log uji coba 1 ditemukan dengan algoritma *alpha*



Gambar 4.30 Data log uji coba 2 ditemukan dengan algoritma *alpha*



Gambar 4.31 Data log PT TPS ditemukan dengan algoritma *alpha*

Setelah menemukan model proses, perlu diketahui *gateway* paralel untuk setiap paralel yang terdapat di model proses. Tabel 4.36 menjelaskan gateway paralel yang didefinisikan oleh masing-masing algoritma berdasarkan data log uji coba 1, data log uji coba 2 dan data log PT TPS.

Tabel 4.36 Perbandingan *gateway* paralel

Data uji	**Algoritma**	***Gateway* Paralel**
Data log uji coba 1	*Modified Time-based Alpha Miner*	B||C : OR D||E : AND
	Alpha	B||C : AND D||E : AND
Data log uji coba 2	*Modified Time-based Alpha Miner*	B||C, C||B : OR D||E, E||D : AND

	Alpha	B||C, C||B : AND D||E, E||D : AND
Data log PT TPS dengan Anomali	*Modified Time-based Alpha Miner*	Seluruh relasi paralel XOR
	Alpha	Seluruh relasi paralel XOR

Selanjutnya evaluasi yang dilakukan adalah dengan membandingkan *trace* dari model proses yang terbentuk dengan *Modified Time-based Alpha Miner* dan algoritma *Alpha* dengan menggunakan data log uji coba 1, data log uji coba 2, dan data log PT TPS. Tabel 4.37 menunjukkan perbedaan *trace* yang ditemukan oleh setiap algoritma.

Tabel 4.37 Perbandingan jumlah *trace*

Data uji	**Algoritma**	**Jumlah *trace* yang ada di data log**	**Jumlah *trace* yang digunakan**
Data log uji coba 1	*Modified Time-based Alpha Miner*	3	2
	Alpha		3
Data log uji coba 2	*Modified Time-based Alpha Miner*	5	3
	Alpha		5
Data log PT TPS	*Modified Time-based Alpha Miner*	20	17
	Alpha		20

BAB V
KESIMPULAN DAN SARAN

Bab ini membahas mengenai kesimpulan yang dapat diambil dari hasil uji coba yang telah dilakukan sebagai jawaban dari rumusan masalah yang dikemukakan. Selain kesimpulan, juga terdapat saran yang ditujukan untuk penelitian dan pengembangan perangkat lunak di masa mendatang.

5.1 Kesimpulan

Dari hasil uji coba data log yang telah dilakukan terhadap sistem yang dikembangkan berdasarkan analisis seluruh metodologi, diambil kesimpulan sebagai berikut:

1. Kondisi untuk paralelisasi aktivitas di dalam proses bisnis harus memenuhi ketentuan yaitu aktivitas independen harus diidentifikasi terlebih dahulu, aktivitas pada entitas simulasi yang sama dijalankan dalam urutan *timestamp*, aktivitas pada entitas simulasi yang sama mengakses item data yang sama dengan cara yang berbeda, waktu dan tempat eksekusi berbeda, *resource* yang menangani setiap aktivitas berbeda, dan *message* setiap departemen ke aktivitas tidak saling bergantung. Pada studi kasus data log PT TPS, Departemen Karantina dan Departemen *Behandle* dapat di paralelkan
2. Modifikasi algoritma *Alpha* menjadi *Modified Time-based Alpha Miner* dilakukan dengan menggunakan *double timestamp* berdasarkan *temporal causal relation* dan *control flow* dalam menemukan model proses sedangkan dalam penentuan *gateway* paralel AND, OR dan XOR dengan cara memberi interval dengan melibatkan rata-rata jumlah relasi paralel, minimum jumlah seluruh relasi *sequence*, dan rata-rata jumlah seluruh relasi *sequence*
3. Penjadwalan (*scheduling)* dengan Sistem Manufaktur Fleksibel dilakukan dengan menggunakan metode *First Come First Serve*. Dari hasil penjadwalan pada bulan Februari di PT TPS, rata-rata RTGC berubah fungsi untuk *container reefer* yaitu 2-3 kali dalam sebulan, untuk *container dry* sebanyak 8-9 kali per bulan dan untuk *container uncontainer* sebanyak 0-1 kali dalam sebulan
4. Optimasi waktu dan biaya dengan menggunakan *Goal programming* dilakukan dengan cara menentukan *goal* atau tujuan dari optimasi, kemudian membuat model matematika sehingga dapat meminimalkan waktu maksimum menjadi waktu rata-rata aktivitas dan biaya yang digunakan lebih kecil daripada biaya maksimum setiap aktivitas.

5.2 Saran

Saran yang diberikan untuk analisis dan pengembangan sistem pada penelitian tesis ini antara lain:

1. Model proses yang dihasilkan *process discovery* pada sistem dapat ditampilkan dalam bentuk *graph* sehingga pengguna lebih mudah melihat modelnya
2. Penentuan *gateway* dalam penelitian ini masih menggunakan pendekatan *heuristics*, diharapkan kedepannya dapat menggunakan pendekatan deterministik
3. Menemukan kriteria lain yang dapat digunakan untuk paralelisasi aktivitas pada proses bisnis
4. Melakukan optimasi waktu dan biaya dengan fungsi dual pada *Goal Programming* dimana waktu dan biaya menjadi *goal*.

DAFTAR PUSTAKA

A.A. Kalenkova, A. Burattin, M. De Leoni, W.M.P. van der Aalst, A. Sperdutti. Discovering High-level BPMN Process Models from Event Log. bpmcenter.org.

Aalst, W. M. (2010). Process Modelling and Analysis. Dalam *Process Mining* (hal. 31-42). Netherlands: Springer.

Aalst, W. M. P. (2010). Process Modelling and Analysis. Dalam *Process Mining* (hal.31-42). Netherlands: Springer.

Agrawal R., Gunopulos D., dan Leyman F. (1998). Mining Process Model from Workflow Logs. *Sixth International Conference Extending Database Technology*, (hal. 469-483).

A.J.M.M. Weijters, J.T.S. Ribeiro. (2010). Flexible Heuristics Miner (FHM). *Beta : Research School for Operations Management and Logistics*.

A.J.M.M. Weijters, W.M.P. van der Aalst, and A.K. Alves de Medeiros. (2006). Process Mining with the Heuristics Miner Algorithm. *BETA Working Paper Series, WP 166.* Eindhoven University of Technology, Eindhoven.

Al-Gahtani, K. S. (2013). Risk Assessment for Scheduling Acceleration. *International Journal of Application or Innovation in Engineering and Management (IJAIEM)*, (hal. 187-193).

Andrea Burattin, Alessandro Sperduti. (2010). Heuristics Miner for Time Intervals. *ESANN 2010 proceedings, European Symposium on Artificial Neural Networks - Computational Intelligence and Machine Learning*, 41-46.

Borja Vazquez-Barreiros, Manuel Mucientes, Manuel Lama. (2015). ProDiGen: Mining complete, precise and minimal structure process models with a genetic algorithm. *Information Sciences, 294*, (hal. 315-333).

Felix Mannhardt, M. De Leoni, H.A. Reijers, van der Aalst. Decision Mining Revisited – Discovering Overlapping Rules. Bpmcenter.org.

Goedertier, S., De Weerdt, J., Martens, D., Vanthienen, J., & Baesens, B. (2011). Process Discovery in Event Log: An Application in the Telecom Industry. *Applied Soft Computing, 11*(2), 1697-1710.

J. Wang, T. He, L. Wen, N. Wu, A.H.M. ter Hofstede, nd J. Su. (2010). A behavioral similarity measure between labeled *Petri net*s based on principal transition *sequence*s. *Proceedings of the 2010 international conference on On the move to meaningful internet systems*, (hal. 394-401).

JongYul Kim, ChangWook Kang, InKeuk Hwang. (2012). A Practical Approach to Project Scheduling : Considering The Potential Quality Loss Cost in The Time-Cost Tradeoff Problem. *International Journal of Project Management*, (hal. 264-272).

M. Song and W.M.P. van der Aalst. (2008). Towards Comprehensive Support for Organizational Mining. *Decision Support Systems, 46*, 300-317.

Mathias, W. (2011). *Business Process Management - Concepts, Languages, Architectures.* Springer.

Nick C. Russell, Aalst, W. M. P., Arthur H,M, ter Hofstede. (2009). Designing a Workflow System Using Coloured *Petri net*s. Dalam J. B. Ed. K. Jensen, *Transactions on Petri nets and Other Models of Concurrency III* (hal. 1-24). Berlin HeidelBerg: Springer.

P. Simin Pulat, Steven J. Horn . (1996). Time-Resource Tradeoff Problem. *IEEE TRANSACTIONS ON ENGINEERING MANAGEMENT*, (hal. 411-417).

Qingtian Zeng, Hua Duan, Cong Liu. (2015). Top-to-down Process Mining from Multi-source Logs Based on Refinement of Petri Nets. 1-22.

Rakesh Agrawal, Dimitrios Gunopulos, Frank Leymann. (1998). Mining Process Models from Workflow Logs.

Riska A. Sutrisnowati, Hyerim Bae, Dongha Lee, Minsoo Kim. (2014). Process Model Discovery based on Activity Lifespan. *International Conference on Technology Innovation and Industrial Management*, (hal. 137-156). Seoul.

Riyanarto Sarno, Widyasari Ayu Wibowo, Kartini, Fitrianing Haryadita, Determining Model Using Non-Linear Heuristics Miner and Control-Flow Pattern, *Telecommunication Computing Electronics and Control (TELKOMNIKA)*, 14 (1), pp. 349-359, 2016

Riyanarto Sarno, Yutika Amelia Effendi, Fitrianing Haryadita. (2016). Modified Time-based Heuristics Miner for Parallel Business Processes. IRECOS.

Riyanarno Sarno, Rahadian Dustrial Dewandono, Tohari Ahmad, Mohammad Farid Naufal, dan Fernandes Sinaga. (2015). Hybrid Association Rule Learning and Process Mining for Fraud Detection. *IAENG International Journal of Computer Science*, (hal. 59-72).

Riyanarto Sarno, Putu Linda Indita Sari, Hari Ginardi, Dwi Sunaryono, Imam Mukhlash. (2013). Decision Mining for Multi Choice Workflow Patterns. *2013 International Conference on Computer, Control, Informatics ad Its Applications* (hal. 337-342). IEEE.

Sanjay Tiwari, Sparsh Johari. (2014). Project Scehduling by Integration of Time Cost Trade-off and Constrained Resource Scheduling. *J. Inst. Eng. India Ser. A* (hal. 37-46). India: Springer.

Sofie De Cnudde, Jan Claes, Geert Poels. (2014). Improving the quality of the Heuristics Miner in Prom 6.2. *Expert Systems with Applications, 41*, (hal. 7678-7690).

Solichul Huda, Riyanarno Sarno, Tohari Ahmad, Heru Agus Santoso. (2014). Identification of Process-based Fraud Patterns in Credit Application. *2nd International Conference on Information and Communication Technology (ICoICT)*, (hal. 84-89).

van der Aalst, W. (2011). *Process Mining - Discovery, Conformance and Enhancement of Business Process.* Netherlands: Springer.

van der Aalst, W. (2013, Oktober). *Process Mining: Beyond Business Intelligence*. Dikutip Juni 2017, 21, dari www.processmining.org

van der Aalst, W., Adriansyah, A., & van Dongen, B. (2011). Causal Nets: A Modeling Language Tailored Towards Process Discovery. *In J.P. Katoen and B. Koenig, editors, 22nd International Conference on Concurrency Theory (CONCUR 2011)*, 28-42.

van der Aalst, B.F. van Dongen. (2012). *Discovering petri nets from event logs, in: Transactions on Petri Nets and Other Models of Concurrency VII.* Springer.

Weske, M. (2011). Business Process Management- Concepts, Languages, Architectures. *Springer*, 128-135.

Zeng, Q., Sun, S. X., Duan, H., Liu, C., & Wang, H. (2013). Cross-organizational collaborative workflow mining from a multi-source log. *Sciencedirect*, (hal. 1280-1301).

(*halaman ini sengaja dikosongkan*)

BIOGRAFI PENULIS

Yutika Amelia Effendi. Anak pertama dari dua bersaudara, kelahiran 14 April 1994 di Kabupaten Solok, Sumatera Barat dari pasangan Jon Effendi dan Fatmawita. Saat ini menetap di Perawang, Tualang, Kab. Siak, Riau. Pendidikan formal di TK YPPI Tualang (1999-2000), SD YPPI Tualang (2000-2006), SMP Negeri 1 Tualang (2006-2009), SMA Negeri 1 Tualang (2009-2012), S1 Teknik Informatika Institut Teknologi Sepuluh Nopember (2012-2016) dan S2 Teknik Informatika di Institut Teknologi Sepuluh Nopember (ITS) pada tahun 2016 hingga sekarang.

Penyuka warna merah ini menyenangi *travelling* dan membaca novel. *An introvert girl and easily distracted by romantic movie and internet world*. Selama masa kuliah, penulis aktif dalam organisasi yang ada di lingkungan kampus ITS dan mengikuti konferensi internasional. Jurnal dan paper penulis yang sudah dipublikasi diantaranya *Modified Time-Based Heuristics Miner for Parallel Business Processes (IRECOS 2016), Non-Linear Optimization of Critical Path Method (ICSITech 2017), Discovering Optimized Process Model using Rule Discovery Hybrid Particle Swarm Optimization (ICSITech 2017), Discovering Process Model from Event Logs by Considering Overlapping Rules (EECSI 2017), SWRL Rules for Identifying Short Loops in Business Process Ontology Model (ICTS 2017), Fraud Detection On Event Log Using Fuzzy Association Rule Learning (ICTS 2017), Determining Model Using Non-Linear Heuristics Miner and Control-Flow Pattern (TELKOMNIKA 2016).* Keterampilan dan penelitian utama adalah process mining, pemodelan proses bisnis, dan Tata Kelola Teknologi Informasi. Penulis dapat dihubungi melalui nomor telepon 082301787770 atau email yutika.effendi@gmail.com.